

# **Básico de Python**

Sergio Pedro Rodrigues Oliveira

# SUMÁRIO

<b>1</b>	<b>Diagrama de estudo</b>	<b>1</b>
<b>2</b>	<b>Variáveis e tipos de dados simples</b>	<b>1</b>
2.1	<code>print()</code>	1
2.2	<code>print()</code> com variáveis	1
2.3	Regras de nomes de variáveis	2
2.4	Concatenando strings	3
2.5	Métodos auxiliares da função <code>print()</code>	3
2.6	Caracteres de escape	5
2.7	Removendo espaços em branco <code>print()</code>	6
2.8	Números	7
2.9	Funções de conversão de tipo	7
2.10	Descobrimo o tipo da variável usando a função <code>type()</code>	8
2.11	Operações básicas	9
2.12	Biblioteca <code>math</code> para ampliar operações matematicas	10
2.13	Operações lógicas básicas	11
2.14	Operadores de identidade	12
2.15	Operações de associação	13
2.16	Comentários	14
2.17	Zen Python	15
<b>3</b>	<b>Listas</b>	<b>16</b>
3.1	Lista	16
3.2	Acessando elementos de uma lista	16
3.3	Alterando, acrescentando e removendo elementos	17
3.3.1	Modificando elementos de uma lista	17
3.3.2	Acrescentando elementos em uma lista	18
3.3.2.1	Concatenando elementos no final de uma lista, método <code>.append()</code>	18
3.3.2.2	Inserindo elementos em uma lista, método <code>.insert()</code>	18
3.3.3	Removendo elementos de uma lista	19
3.3.3.1	Instrução <code>del</code>	19
3.3.3.2	Método <code>.pop()</code>	19
3.3.3.3	Método <code>.remove()</code>	20
3.4	Organizando uma lista	22
3.4.1	Método <code>.sort()</code>	22
3.4.2	A função <code>sorted()</code>	23
3.4.3	Método <code>.reverse()</code>	24
3.5	Descobrimo o tamanho de uma lista - <code>len()</code>	25

<b>4</b>	<b>Trabalhando com listas</b>	<b>26</b>
4.1	Percorrendo uma lista inteira com um laço . . . . .	26
4.2	Erros comuns de indentação . . . . .	26
4.3	Listas numéricas . . . . .	27
4.3.1	Gerando série de números com a função <code>range()</code> . . . . .	27
4.3.2	Usando <code>range()</code> para gerar uma lista - <code>list()</code> . . . . .	27
4.3.3	Estatística simples com lista de números . . . . .	28
4.4	<code>list comprehensions</code> . . . . .	29
4.5	Trabalhando com parte de uma lista . . . . .	30
4.5.1	Fatiando uma lista . . . . .	30
4.5.2	Percorrendo uma fatia com um laço - <code>for</code> . . . . .	32
4.5.3	Copiando uma lista . . . . .	33
4.6	Tuplas . . . . .	35
4.6.1	Definindo uma tupla . . . . .	35
4.6.2	Percorrendo todos os valores de uma tupla com um laço . . . . .	35
4.6.3	Sobrescrevendo uma tupla . . . . .	36
<b>5</b>	<b>Estatística básica</b>	<b>37</b>
5.1	Teoria . . . . .	37
5.2	Preparação dos dados (sumariar dados coletados) . . . . .	39
5.2.1	Variável Quantitativa Discreta . . . . .	40
5.2.2	Variável Quantitativa Contínua . . . . .	41
5.2.3	Variáveis Qualitativas . . . . .	47
5.3	Medidas de posição . . . . .	48
5.3.1	Média Aritmética (Simples e Ponderada) . . . . .	49
5.3.2	Mediana ( $md(x)$ ) . . . . .	50
5.3.2.1	Mediana Discreta . . . . .	50
5.3.2.2	Mediana Contínua . . . . .	50
5.3.3	Moda . . . . .	51
5.3.4	Separatrizes . . . . .	53
5.4	Medidas de dispersão . . . . .	55
5.4.1	Amplitude Total ( $A_T$ ) . . . . .	55
5.4.2	Desvio . . . . .	56
5.4.2.1	Desvio Absoluto ( $D$ ) . . . . .	56
5.4.2.2	Desvio Absoluto Médio ( $dm$ ) . . . . .	57
5.4.3	Variância ( $\sigma^2$ ou $S^2$ ) . . . . .	58
5.4.4	Desvio-padrão ( $\sigma$ ou $S$ ) . . . . .	60
5.4.4.1	Variância x Desvio-padrão . . . . .	60
5.4.4.2	Desvio-padrão (Populacional e Amostral) . . . . .	60
5.4.5	Coefficiente de Variação ( $CV$ ) . . . . .	61
5.4.5.1	Teoria . . . . .	61
5.4.5.2	Cálculo do Coeficiente de Variação . . . . .	61

<b>6</b>	<b>Análise Estatística</b>	<b>62</b>
<b>7</b>	<b>Instruções IF</b>	<b>63</b>
7.1	Testes condicionais	63
7.2	Operações lógicas	63
7.3	Testando várias condições	65
7.3.1	Testando várias condições lógicas - <b>AND</b>	65
7.3.2	Testando várias condições lógicas - <b>OR</b>	65
7.4	Verificando se um valor está em uma lista - <b>IN</b>	67
7.5	Verificando se um valor não está em uma lista - <b>NOT IN</b>	67
7.6	Expressões booleanas	67
7.7	Instruções IF	68
7.7.1	Instruções <b>if</b> simples	68
7.7.2	Instruções <b>if-else</b>	69
7.7.3	Sintaxe <b>if-elif-else</b>	69
7.7.4	Usando vários blocos <b>elif</b>	70
7.7.5	Omitindo o bloco <b>else</b>	71
7.7.6	Testando várias condições	72
7.8	Usando instruções <b>if</b> com listas	73
7.9	Verificando itens especiais	73
7.10	Verificando se uma lista não está vazia	73
7.11	Usando várias listas	73

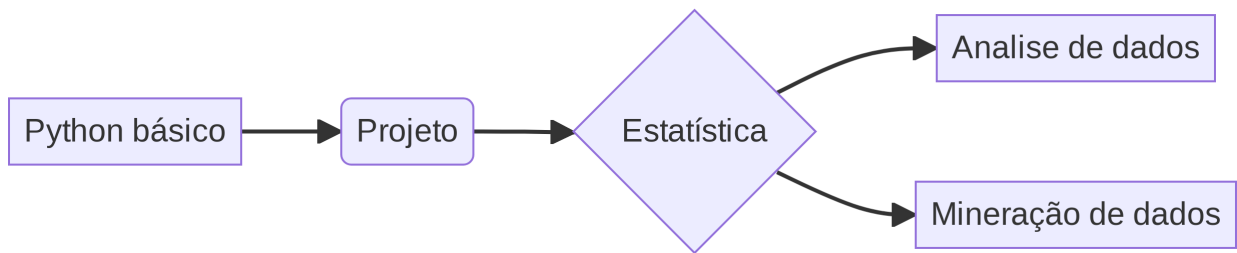
## LISTA DE FIGURAS

1	Fluxograma da estatística descritiva. . . . .	37
2	Tipos de variáveis. . . . .	39
3	Distribuição tabular quantitativa discreta. . . . .	40
4	Distribuição de frequências em classes. . . . .	41
5	Intervalo de classes, distribuição de frequências quantitativa continua. . . . .	42
6	Distribuição frequências quantitativa continua, premissas. . . . .	42
7	Tabela de distribuição de frequência quantitativa continua. . . . .	46
8	Exemplo de dispersão com heterogeneidade e homogeneidade. . . . .	55
9	Tabela verdade do operador <b>AND</b> . . . . .	65
10	Tabela verdade do operador <b>OR</b> . . . . .	66

## LISTA DE TABELAS

1	Caracteres de escape . . . . .	5
2	Principais tipos de dados . . . . .	7
3	Funções de conversão de tipo . . . . .	7
4	Operações básicas . . . . .	9
5	Algumas operações da biblioteca <code>math</code> . . . . .	10
6	Operações Lógicas Básicas . . . . .	11
7	Operadores identidade . . . . .	12
8	Operadores de associação . . . . .	13
9	Estatística simples . . . . .	28
10	Medidas de posição, bibliotecas <code>python</code> . . . . .	28
11	Medidas de dispersão, bibliotecas <code>python</code> . . . . .	28
12	Operações Lógicas . . . . .	63
13	Operações Lógicas Exemplos . . . . .	64

## 1 Diagrama de estudo



## 2 Variáveis e tipos de dados simples

### 2.1 print()

Print é uma função que exibe uma string na tela.

Exemplo:

```
print("string")
```

string

### 2.2 print() com variáveis

Podemos usar a função `print()` para imprimir uma variável string.

Exemplo:

```
message = "Hello world!"  
print(message)
```

Hello world!

## 2.3 Regras de nomes de variáveis

Regras ou diretrizes para usar variáveis em Python.

- Nomes de variáveis deve conter apenas letras, números e underscores. Podemos começar a variável com letra ou underscore, mas nunca com um número.
- Espaços não são permitidos em nomes de variáveis, mas underscores podem ser usados para separar palavras.
- Evite usar palavras reservadas e nome de funções em Python como nome de variáveis.
- Nomes de variáveis devem ser concisos, porém descritivos.
- Tome cuidado ao usar a letra l e a letra maiúscula O, pois podem ser confundidas com os números 1 e 0.



## 2.4 Concatenando strings

Podemos usar o simbolo de (+) para combinar strings (concatenar).

Exemplo:

```
first_name = "ada"
last_name = "lovelace"
full_name = first_name + " " + last_name
print("Hello, " + full_name.title() + "!")
```

Hello, Ada Lovelace!

Os espaços em branco entre aspas servem para criar espaços na string.

## 2.5 Métodos auxiliares da função print()

### 1. .title()

Coloca apenas as primeiras letras em maiúsculas de cada palavra e o resto em minúscula.

Exemplo:

```
full_name = "ada lovelace"
print(full_name.title())
```

Ada Lovelace

### 2. .upper()

Coloca todas as letras em maiúsculas.

Exemplo:

```
full_name = "ada lovelace"
print(full_name.upper())
```

ADA LOVELACE

### 3. .lower()

Coloca todas as letras em minúsculas. O método `.lower()` é particularmente útil para armazenar dados. Converter os dados em minúscula antes de armazenar.

Exemplo:

```
full_name = "ada lovelace"  
print(full_name.lower())
```

```
ada lovelace
```

## 2.6 Caracteres de escape

Podemos inserir alguns caracteres de escape no texto para executar alguma ação, como pular linha, gerar tabulação e etc. Alguns caracteres podem ser vistos na [Table 1](#).

Todos os caracteres de escape começam com barra(\) + complemento.

Table 1: Caracteres de escape

Caracteres de escape	Descrição
\t	Gera tabulação (tab).
\n	Gera quebra de linha.

Exemplo:

```
print("Language:\nPython\nJava\nC\nJavaScript")
```

```
Language:
Python
Java
C
JavaScript
```

## 2.7 Removendo espaços em branco print()

### 1. .rstrip()

Remove espaço em branco do lado direito.

Exemplo:

```
favorite_language = 'python '  
favorite_language.rstrip()
```

'python'

### 2. .lstrip()

Remove espaço em branco do lado esquerdo.

Exemplo:

```
favorite_language = ' python'  
favorite_language.lstrip()
```

'python'

### 3. .strip()

Remove os espaços em branco dos dois lados ao mesmo tempo.

Exemplo:

```
favorite_language = ' python '  
favorite_language.strip()
```

'python'

- Os metodos usados não removem os espaços em branco em definitivo, para remover em definitivo é necessario armazenar o valor novo na variável.

```
favorite_language = ' python '  
favorite_language = favorite_language.strip()  
favorite_language
```

'python'

## 2.8 Números

A linguagem Python faz tipagem automática (dinâmica), tipa a variável de acordo com o uso. E o Python contém uma tipagem forte, não faz converção automática do tipo de uma variável para executar uma ação (operação).

Em resumo, python tem é uma linguagem de tipagem dinâmica e forte.

Os principais tipos de dados no Python são estão presentes na Table 2.

Table 2: Principais tipos de dados

Nome	Abreviação	Descrição
Inteiro	int	Números inteiros
Ponto flutuante	float	Números com ponto decimal

## 2.9 Funções de conversão de tipo

Podemos converter variáveis para determinado tipo especificado usando funções de conversão de tipo, como pode ser observado na Table 3.

Converter uma variável não é permanente, a não ser que a ação seja armazenada na variável explicitamente.

Table 3: Funções de conversão de tipo

Tipo para converter	Função	Descrição
int	<code>int()</code>	Converte variável para o tipo inteiro(int)
float	<code>float()</code>	Converte variável para o tipo float
string	<code>str()</code>	Converte variável para o tipo string

A função `str()` é deveras importante, pois pode auxiliar na função `print()`. A função `print()` só imprime na tela variáveis string, sendo assim, precisamos converter as variáveis de outros tipos para string (pelo menos, momentaneamente), para cumprir essa condição.

Exemplo:

```
age = 23
print("Happy " + str(age) + "rd Birthday!")
```

Happy 23rd Birthday!

## 2.10 Descobrindo o tipo da variável usando a função `type()`

Podemos usar a função `type()` para descobrir o tipo de determinada variável.

```
age = 23  
print (type(age))
```

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

É uma **boa prática** usar a função `type()`, para conferir o tipo da variável, antes de manipular alguma variável. Assim o programador terá o controle sobre as variáveis que esta trabalhando. Essa boa prática evita erros.

Também é uma **boa prática**, ao identificar/observar um erro, conferir os tipos das variáveis envolvidas. É um dos erros mais comuns: erro de tipagem.

## 2.11 Operações básicas

A Table 4 apresenta as principais operações básicas do python.

Table 4: Operações básicas

Operação	Símbolo	Exemplo
Soma	+	$2+2=4$
Subtração	-	$3-2=1$
Multiplicação	*	$2*3=6$
Divisão	/	$5/4=1.25$
Divisão inteira	//	$5//4=1$
Resto da divisão (módulo)	%	$10\%8=2$
Potência	**	$3**2=9$
Raiz	**	$4**0.5=2$

## 2.12 Biblioteca `math` para ampliar operações matemáticas

Podemos usar o pacote `math` para ampliar as funções matemáticas do Python (básicas, trigonométricas e estatísticas). A Table 5 apresenta as principais funções básicas da biblioteca `math`.

Table 5: Algumas operações da biblioteca `math`

Operação	Símbolo	Exemplo
Soma	<code>math.add(x,y)</code>	<code>math.add(2,2) = (2+2)=4</code>
Subtração	<code>math.subtract(x,y)</code>	<code>math.subtract(2,2) = (2-2)=0</code>
Raiz quadrada	<code>math.sqrt()</code>	<code>math.sqrt(4)=2</code>
Potência	<code>math.pow(x,y)</code>	<code>math.pow(2,3) = (2**3)=8</code>
Seno	<code>math.sin()</code>	<code>math.sin()</code> , retorna um ângulo em radianos.
Cosseno	<code>math.cos()</code>	<code>math.cos()</code> , retorna um ângulo em radianos.
Tangente	<code>math.tan()</code>	<code>math.tan()</code> , retorna um ângulo em radianos.
potencia de Euler	<code>math.exp(x)</code>	<code>math.exp(x) = math.pow(math.e**x)</code>
Logaritmo natural, ou log neperiano	<code>math.log(x)</code>	<code>math.log(2)=0.69</code>
Logaritmo	<code>math.log(x[,base])</code>	<code>math.log(2,10)=0.3</code>

Para converter o ângulo para radianos podemos usar a função `math.radians()`.

```
import math
#Seno do ângulo de 45°
#Resultado em Radianos
print(str(math.sin(math.radians(45))))
```

0.7071067811865475

Para converter de radiano para grau podemos usar a função `math.degrees()`.

```
import math
#Seno do ângulo de 45°
#Resultado em ângulo
print(str(math.degrees(math.sin(math.radians(45)))))
```

40.51423422706977



## 2.13 Operações lógicas básicas

A Table 6 apresenta as principais operações lógica básica do python. As operações lógicas retornam `True` ou `False`.

Table 6: Operações Lógicas Básicas

Operação	Nome	Função	Exemplo
<code>==</code>	Igual a	Varifica se um valor é igual ao outro.	<code>1==1 = True</code>
<code>!=</code>	Diferente de	Varifica se um valor é diferente ao outro.	<code>1!=2 = True</code>
<code>&gt;</code>	Maior que	Varifica se um valor é maior que outro.	<code>5&gt;1 = True</code>
<code>&gt;=</code>	Maior ou igual	Varifica se um valor é maior ou igual a outro.	<code>5&gt;=5 = True</code>
<code>&lt;</code>	Menor que	Varifica se um valor é menor que outro.	<code>1&lt;5 = True</code>
<code>&lt;=</code>	Menor ou igual	Varifica se um valor é menor ou igual a outro.	<code>1&lt;=4 = True</code>
<code>and</code>	E	Retorna <code>True</code> se ambas as afirmações forem verdadeiras.	<code>(1==1) and (4&lt;5)</code>
<code>or</code>	Ou	Retorna <code>True</code> se uma das afirmações for verdadeiras.	<code>(1==1) or (2&lt;1)</code>
<code>not</code>	Negação	Retorna <code>Falso</code> se o resultado for verdadeiro, ou o contrario.	<code>not (1==1) = False</code>

## 2.14 Operadores de identidade

Os operadores de identidade, Table 7, são utilizados para comparar objetos, se os objetos testados referenciam o mesmo objeto.

Table 7: Operadores identidade

Operador	Definição
is	Retorna <b>True</b> se ambas as variáveis são o mesmo objeto.
is not	Retorna <b>True</b> se ambas as variáveis não são o mesmo objeto.

Exemplo de operações de identidade:

```
lista = [1,2,3]
outra_lista = [1,2,3]
recebe_lista = lista

print(f"São o mesmo objeto: {lista is outra_lista}")
```

São o mesmo objeto: False

```
lista = [1,2,3]
outra_lista = [1,2,3]
recebe_lista = lista

print(f"São o mesmo objeto: {lista is recebe_lista}")
```

São o mesmo objeto: True

## 2.15 Operações de associação

Os operadores de associação, Table 8, servem para verificar se determinado objeto esta **associado** ou **pertence** a determinada estrutura de dados.

Table 8: Operadores de associação

Operação	Função
in	Retorna <b>True</b> caso valor seja encontrado na sequência.
not in	Retorna <b>True</b> caso valor não seja encontrado na sequência.

Exemplos de operações de associação:

```
lista = ["Python", 'Academy', "Operadores", 'Condições']  
print('Python' in lista)
```

True

```
lista = ["Python", 'Academy', "Operadores", 'Condições']  
print('SQL' not in lista)
```

True

## 2.16 Comentários

Um comentário permite escrever notas em seus programas em linguagem natural. Em Python, o caractere sustenido (#) indica um comentário. Tudo que vier depois de um caractere sustenido em seu código será ignorado pelo interpretador Python.

**Boas práticas** em comentários:

1. Explicar o que o código deve fazer.
2. Como faz para funcionar.

## 2.17 Zen Python

É um guia de **boas práticas**.

```
import this
```

Principais pontos:

1. Bonito é melhor do que feio.
2. Simples é melhor que complexo.
3. Complexo é melhor que complicado.
4. Legibilidade conta.
5. Deve haver uma - e, de preferência, apenas uma - maneira óbvia de fazer algo.
6. Agora é melhor que nunca.

## 3 Listas

### 3.1 Lista

Uma lista é uma coleção de itens em uma ordem em particular. Os colchetes([]) indicam uma lista e os elementos individuais de uma lista são separados por vírgula. [ver [1](#), p. 71]

Exemplo:

```
bicycles = ['trek','cannondale','redline','specialized']
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

### 3.2 Acessando elementos de uma lista

Podemos acessar a qualquer item de uma lista informando a posição, ou índice. As posições de uma lista começam no 0, e não no 1.

Para acessar um elemento de uma lista, informamos o nome da lista seguido do índice do item entre colchetes.

Exemplo:

```
#Acessando o primeiro item da lista
bicycles = ['trek','cannondale','redline','specialized']
print(bicycles[0].title())
```

Trek

Para acessar a lista de trás pra frente podemos usar a posição invertida seguida do símbolo de menos na frente. Sendo assim, a posição do último item é -1, do penúltimo -2 e assim sucessivamente.

Exemplo:

```
#Acessando o último item da lista
bicycles = ['trek','cannondale','redline','specialized']
print(bicycles[-1].title())
```

Specialized

### 3.3 Alterando, acrescentando e removendo elementos

Dado que a lista é um elemento dinâmico (pode, e provavelmente ocorrerá, de sofrer modificações com o uso), este tópico comentará os principais formas de modificação de listas.

#### 3.3.1 Modificando elementos de uma lista

Para alterar um elemento que você quer modificar, use o nome da lista seguido do índice do elemento que quer modificar, e então forneça um novo valor.

```
#Alterando o item 1 da lista (índice 0)
motorcycles = ['honda','yamaha','suzuki']
motorcycles[0] = 'ducati'
print(motorcycles)
```

```
['ducati', 'yamaha', 'suzuki']
```

### 3.3.2 Acrescentando elementos em uma lista

Existem diversas formas de adicionar elementos a uma lista:

#### 3.3.2.1 Concatenando elementos no final de uma lista, método `.append()`

Adiciona um novo elemento no final da lista usando o método `.append()`.

Exemplo:

```
#Adicionando elemento ao final da lista
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles.append('ducati')
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki', 'ducati']
```

#### 3.3.2.2 Inserindo elementos em uma lista, método `.insert()`

Este método insere um elemento em determinada posição da lista, usando o método `.insert(índice, elemento)`.

Exemplo:

```
#Adicionando um item na segunda posição da lista (índice 1)
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles.insert(1, 'ducati')
print(motorcycles)
```

```
['honda', 'ducati', 'yamaha', 'suzuki']
```



### 3.3.3 Removendo elementos de uma lista

Os métodos para remover um item, ou um conjunto de itens, de uma lista.

#### 3.3.3.1 Instrução `del`

Se a posição do item que você quer remover de uma lista for conhecida, a instrução `del` remove (deleta) um item em qualquer determinada posição. Depois de removido (deletado) não podemos mais acessar o valor, quando usado a instrução `del`.

```
# Remover (deletar) primeiro item da lista, índice 0
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)

del motorcycles[0]
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']
['yamaha', 'suzuki']
```

#### 3.3.3.2 Método `.pop()`

Existem duas formas de usar o método `.pop()`:

1. `.pop()`

As vezes há necessidade de usar o valor de um item depois de removê-lo de uma lista. O método `.pop()` remove o **último** item de uma lista, mas permite que você trabalhe com esse item depois da remoção.

Remove o primeiro item de uma pilha, ou seja, o último item de uma lista.

Para usarmos o item removido é necessário, salva-lo numa variável.

Exemplo:

```
# Uso do método .pop()
# Removendo último item da lista e
# Trabalhando com o item removido.
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)

pop_motorcycle = motorcycles.pop()
```

```
print(motorcycles)
print(pop_motorcycle)
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha']
suzuki
```

## 2. .pop(índice)

Podemos usar o `.pop()` para remover um item em qualquer posição em uma lista, se incluirmos o índice do item que você deseja remover entre parênteses.

Exemplo:

```
# Uso do método .pop()
# Removendo o segundo item da lista e
# Trabalhando com o item removido.
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)

pop_motorcycle = motorcycles.pop(1)
print(motorcycles)
print(pop_motorcycle)
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'suzuki']
yamaha
```

### 3.3.3.3 Método `.remove()`

Remove um item de acordo com o valor. É usado quando sabemos o valor do item, mas não a posição.

O método `.remove()` apaga apenas a primeira ocorrência do valor especificado. Para apagar mais de uma ocorrência será necessário o uso de um laço, para cada ocorrência.

Exemplo:

```
# Uso do método .remove()
# Removendo um item da lista pelo valor
motorcycles = ['honda', 'yamaha', 'ducati']
print(motorcycles)
```

```
too_expensive = 'ducati'  
motorcycles.remove(too_expensive)  
print(motorcycles)
```

```
['honda', 'yamaha', 'ducati']  
['honda', 'yamaha']
```

## 3.4 Organizando uma lista

Dado que com frequência, as listas são organizadas numa ordem imprevisível, se torna necessário organizar as informações em uma ordem particular. O Python tem mecanismos para organizar listas. São eles:

### 3.4.1 Método `.sort()`

Ordena uma lista em ordem alfabética, ou alfabética inversa.

Para ordenar uma lista em ordem alfabética inversa, basta passar o argumento `reverse = True` para o método `.sort()`.

Uma vez ordenada pelo método `.sort()` a lista não retorna a ordem original (ordenação permanente).

Exemplo:

```
# Ordenando a lista cars usando o método .sort()
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
cars.sort()
print(cars)
cars.sort(reverse=True)
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']
['audi', 'bmw', 'subaru', 'toyota']
['toyota', 'subaru', 'bmw', 'audi']
```

### 3.4.2 A função sorted()

A função `sorted()` ordena uma lista de forma temporaria, não altera a lista original, em ordem alfabética. Ou seja, a lista volta a forma original ao final do uso da função.

Assim como no método `.sort()`, podemos ordenar a lista em ordem alfabética inversa adicionando o argumento `reverse=True`.

Exemplo:

```
# Ordenando temporariamente a lista cars usando a função sorted()
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
print(sorted(cars))
print(sorted(cars, reverse=True))
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']
['audi', 'bmw', 'subaru', 'toyota']
['toyota', 'subaru', 'bmw', 'audi']
['bmw', 'audi', 'toyota', 'subaru']
```

### 3.4.3 Método `.reverse()`

Para inverter a ordem original de uma lista, podemos usar o método `.reverse()`.

O método `.reverse()` não organiza a lista em ordem alfabética inversa, o método inverte a lista original.

O método `.reverse()` ordena de forma permanente a lista, porém se usarmos o método novamente, teremos a lista original. Logo, é fácil reverter o uso do método `.reverse()`.

Exemplo:

```
# Método .reverse() para inverte, de modo permanente, a ordem da lista.
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
cars.reverse()
print(cars)
cars.reverse()
print(cars)
```

```
['bmw', 'audi', 'toyota', 'subaru']
['subaru', 'toyota', 'audi', 'bmw']
['bmw', 'audi', 'toyota', 'subaru']
```

### 3.5 Descobrindo o tamanho de uma lista - len()

Podemos descobrir o tamanho de uma lista usando a função `len()`.

Exemplo:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']  
len(cars)
```

4

## 4 Trabalhando com listas

### 4.1 Percorrendo uma lista inteira com um laço

Podemos usar um laço `for` para percorrer toda uma lista, podendo assim entre outras coisas, efetuar tarefas em cada item da lista.

A estrutura básica do `for` é:

```
for variável_nova in lista :  
    tarefas
```

O laço diz para a cada iteração pegar um elemento da lista e armazenar na nova variável, e executar uma tarefa a cada iteração. Toda tarefa indentada depois dos dois pontos é considerada dentro do laço.

No Python o `for`, usa indentação para determinar o que esta dentro do laço.

Qual quer linha após o laço que não for indentada é considerada fora do laço.

Exemplo:

```
#Executando um laço com base numa lista  
magicians = ['alice', 'david', 'carolina']  
for magician in magicians:  
    print(magician)
```

```
alice  
david  
carolina
```

### 4.2 Erros comuns de indentação

- Esquecer de indentar.
- Esquecer de indentar linhas adicionais do laço.
- Indentação desnecessaria.
- Indentando desnecessariamente após o laço.
- Esquecer os dois-pontos do laço `for`.



## 4.3 Listas numéricas

### 4.3.1 Gerando série de números com a função `range()`

A função `range()` é usada para gerar uma série de números, de uma determinada sequência numérica.

A função `range()` faz o Python começar a contar no primeiro valor definido (limite inferior) e parar quando atingir o segundo valor definido (limite superior). Como o `for` para no segundo valor, a saída não conterá o valor final. Também podemos definir um intervalo, pulando alguns valores.

Estrutura da função `range()`:

```
range(limite_inferior, limite_superior, intervalo)
```

Exemplo:

```
for value in range(1,5):  
    print(value)
```

```
1  
2  
3  
4
```

### 4.3.2 Usando `range()` para gerar uma lista - `list()`

Podemos usar para criar uma lista de números, combinando a função `range()`, que gera uma série numérica, com a função `list()`, que cria um lista.

Exemplo:

```
numbers = list(range(1,6))  
print(numbers)
```

```
[1, 2, 3, 4, 5]
```

Exemplo 2:

```
numbers = list(range(2,11,2))  
print(numbers)
```

```
[2, 4, 6, 8, 10]
```

### 4.3.3 Estatística simples com lista de números

As principais funções estatísticas estão contidas na [Table 9](#).

Table 9: Estatística simples

Funções	Descrição
<code>min()</code>	Retorna o valor mínimo.
<code>max()</code>	Retorna o valor máximo.
<code>sum()</code>	Somatório.

As principais bibliotecas auxiliares de funções estatísticas são:

1. `math`
2. `numpy` as `np`
3. `statistics`
4. `Pandas` as `pd`

Medidas de posição utilizando bibliotecas python, [Table 10](#).

Table 10: Medidas de posição, bibliotecas python

Funções	Descrição
<code>np.mean()</code>	Média aritmética
<code>statistics.median()</code>	Mediana
<code>statistics.mode()</code>	Moda
<code>np.quantiles(array, 0.5)</code>	Quartil
<code>np.percentile(array, 50)</code>	Percentil

Medidas de dispersão utilizando bibliotecas python, [Table 11](#).

Table 11: Medidas de dispersão, bibliotecas python

Funções	Descrição
<code>pd.var()</code>	Variância
<code>pd.std()</code>	Desvio-padrão
<code>pd.mad()</code>	Desvio absoluto
<code>pd.cov()</code>	Covariância
<code>pd.corr()</code>	Correlação

## 4.4 list comprehensions

List comprehensions é uma forma de criar listas já acoplando o laço for nelas, deixando o código mais enxuto.

Sintaxe:

```
nome_lista = [expressão_calculada_do_for for variável in range()]
```

Exemplo:

```
squares = [value ** 2 for value in range(1,11)]  
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

## 4.5 Trabalhando com parte de uma lista

Neste tópico vamos trabalhar com um grupo de itens de uma lista, no Python é chamado de *fatia* (de uma lista).

### 4.5.1 Fatiando uma lista

#### 1. Fatia simples

Para criar uma fatia, especifique o índice do primeiro e o último elemento com os quais você deseja trabalhar.

O Python para em um item antes do segundo índice (índice final) especificado.

Exemplo:

```
#Exibindo os 3 primeiros elementos de uma lista.  
players = ["charles","martina","michael","florence","eli"]  
print(players[0:3])  
#Serão exibidos os itens na posição 0, 1 e 2.
```

```
['charles', 'martina', 'michael']
```

#### 2. Delimitando início e fim da fatia.

Podemos começar de qualquer índice.

Exemplo:

```
#Exibindo do segundo ao quarto item.  
players = ["charles","martina","michael","florence","eli"]  
print(players[1:4])
```

```
['martina', 'michael', 'florence']
```

### 3. Omitindo índices

Se omitirmos o primeiro índice, o Python começará do índice 0 (início). De maneira análoga, se omitirmos o segundo índice (índice final), o Python terminará no último item.

Exemplo:

```
#Exibindo os 2 primeiros elementos de uma lista.  
players = ["charles","martina","michael","florence","eli"]  
print(players[:2])
```

```
['charles', 'martina']
```

### 4. índice negativo

O índice negativo devolve um elemento a determina distância do final da lista. Assim podemos exibir qualquer fatia a partir do final da lista.

Exemplo:

```
#Exibindo os 3 últimos elementos de uma lista.  
players = ["charles","martina","michael","florence","eli"]  
print(players[-3:])
```

```
['michael', 'florence', 'eli']
```

#### 4.5.2 Percorrendo uma fatia com um laço - for

Podemos usar uma fatia em um laço `for` se quisermos percorrer um subconjunto de elementos de uma lista.

Exemplo:

```
players = ["charles","martina","michael","florence","eli"]
print("Here are the first three players on my team:")
for player in players[:3]:
    print(player.title())
```

Here are the first three players on my team:

Charles

Martina

Michael

### 4.5.3 Copiando uma lista

Vamos explorar o modo de copiar uma lista e analisar uma situação em que copiar uma lista é útil.

1. Copiando uma lista inteira, usando *fatia*.

Podemos criar uma fatia que inclua a lista inteira, omitindo o primeiro e segundo índices.

Exemplo:

```
#Usamos o método de fatia para copiar listas.
my_foods = ["pizza","falafel","carrot cake"]
friend_foods = my_foods[:]

print("My favorite food are:")
print(my_foods)

print("\nMy friend's favorite food are:")
print(friend_foods)
```

```
My favorite food are:
['pizza', 'falafel', 'carrot cake']
```

```
My friend's favorite food are:
['pizza', 'falafel', 'carrot cake']
```

Ambas as listas `my_foods` e `friend_foods`, contém os mesmos elementos, porém são listas diferentes. Ao modificarmos uma delas a outra não é modificada automaticamente, por serem listas diferentes.

2. Variáveis que apontam para mesma lista.

Se ao invés de copiarmos uma fatia de uma lista para a outra, mesmo que seja a lista inteira, definirmos que uma variável é igual a outra, nesse caso criamos duas variáveis que apontam para a mesma lista. Ou seja, se modificarmos qualquer uma das listas, a outra é automaticamente modificada, pois ambas são a mesma lista.

Exemplo:

```
#Ambas variáveis apontam para a mesma lista.
my_foods = ["pizza","falafel","carrot cake"]
friend_foods = my_foods

friend_foods.append("ice cream")

print("My favorite food are:")
print(my_foods)

print("\nMy friend's favorite food are:")
print(friend_foods)
```

```
My favorite food are:
['pizza', 'falafel', 'carrot cake', 'ice cream']
```

```
My friend's favorite food are:
['pizza', 'falafel', 'carrot cake', 'ice cream']
```



## 4.6 Tuplas

Tuplas são listas em que os itens não são criadas para mudar (listas imutáveis).

### 4.6.1 Definindo uma tupla

Uma tupla se parece com uma lista, exceto por usar parênteses no lugar de colchetes.

Sintaxe:

```
tuplas = (valor_1, valor_2, valor_3, ...)
```

Exibimos cada elemento de uma tupla com a mesma sintaxe que usamos para acessar elementos de uma lista.

Exemplo:

```
dimensions = (200, 50)
print(dimensions[0])
```

200

Se tentarmos alterar algum elemento de uma tupla, será retornado um erro de tipo.

### 4.6.2 Percorrendo todos os valores de uma tupla com um laço

Podemos percorrer uma tupla usando um laço `for`, da mesma forma que uma lista.

Exemplo:

```
dimensions = (200, 50)
for dimension in dimensions:
    print(dimension)
```

200

50

### 4.6.3 Sobrescrevendo uma tupla

Não é possível modificar os elementos de uma tupla. Retornaria um erro de tipo.

Esse tipo de operação não funcionaria:

```
tupla[0] = valor_novo
```

Porém é possível sobrescrever a tupla, imputando novos valores a variável.

Exemplo:

```
#Sobrescrevendo uma tupla
dimensions = (200,50)
print("Original dimensions:")
for dimension in dimensions:
    print(dimension)

dimensions = (400,100)
print("\nModified dimensions:")
for dimension in dimensions:
    print(dimension)
```

Original dimensions:

200

50

Modified dimensions:

400

100

## 5 Estatística básica

### 5.1 Teoria

- Definição de Estatística:

A Estatística de uma maneira geral compreende aos métodos científicos para COLETA, ORGANIZAÇÃO, RESUMO, APRESENTAÇÃO e ANÁLISE de Dados de Observação (Estudos ou Experimentos), obtidos em qualquer área de conhecimento. A finalidade é a de obter conclusões válidas para tomada de decisões.

- Estatística Descritiva

Parte responsável basicamente pela COLETA e SÍNTESE (Descrição) dos Dados em questão.

Disponibiliza técnicas para o alcance desses objetivos. Tais Dados podem ser provenientes de uma AMOSTRA ou POPULAÇÃO.

- Estatística Inferencial

É utilizada para tomada de decisões a respeito de uma população, em geral fazendo uso de dados de amostrais.

Essas decisões são tomadas sob condições de INCERTEZA, por isso faz-se necessário o uso da TEORIA DA PROBABILIDADE.

- O fluxograma da estatística descritiva pode ser esposto da seguinte forma:



Figure 1: Fluxograma da estatística descritiva.

- A representação tabular (**Tabelas de Distribuição de Frequências**) deve conter:
  - Cabeçalho  
Deve conter o suficiente para que as seguintes perguntas sejam respondidas “**o que?**” (Relativo ao fato), “**onde?**” (Relativo ao lugar) e “**quando?**” (Correspondente à época).
  - Corpo  
É o lugar da Tabela onde os dados serão registrados. Apresenta colunas e sub colunas.
  - Rodapé  
Local destinado à outras informações pertinentes, por exemplo a Fonte dos Dados.
- População e Amostras:
  - População  
É o conjunto de todos os itens, objetos ou pessoas sob consideração, os quais possuem pelo menos uma característica (variável) em comum. Os elementos pertencentes à uma População são denominados “Unidades Amostrais”.
  - Amostras  
É qualquer subconjunto (não vazio) da População. É extraída conforme regras pré-estabelecidas, com a finalidade de obter “estimativa” de alguma característica da População.

- Tipos de variáveis

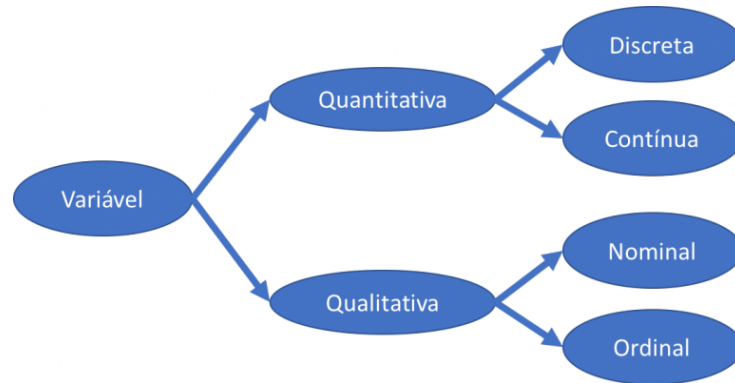


Figure 2: Tipos de variáveis.

- *Qualitativo nominal*  
Não possuem uma ordem natural de ocorrência.
- *Qualitativo ordinal*  
Possuem uma ordem natural de ocorrência.
- *Quantitativo discreta*  
Só podem assumir valores inteiros, pertencentes a um conjunto finito ou enumerável.
- *Quantitativo contínua*  
Podem assumir qualquer valor em um determinado intervalo da reta dos números reais.

## 5.2 Preparação dos dados (sumariar dados coletados)

- Frequência (conceito)  
É a quantidade de vezes que um valor é observado dentro de um conjunto de dado.
- Distribuição em frequências
  - A distribuição tabular é denominada: “Tabela de Distribuição de Frequências”.
  - Podemos separar em 3 modelos de distribuição tabular:
    - \* Variável Quantitativa Discreta.
    - \* Variável Quantitativa Contínua.
    - \* Variáveis Qualitativas.

### 5.2.1 Variável Quantitativa Discreta

- Passos da preparação dos dados:
  - 1º Passo - **DADOS BRUTOS:**  
Obter os dados da maneira que foram coletados.
  - 2º Passo - **ROL:**  
Organizar os DADOS BRUTOS em uma determinada ordem (crescente ou decrescente).
  - 3º Passo - **CONSTRUÇÃO TABELA:**  
Na primeira coluna são colocados os valores da variável, e nas demais as respectivas frequências.
  - Frequência absoluta simples.  
Nº de vezes que cada valor da variável se repete.
- Principais campos da **distribuição tabular de variáveis quantitativas discretas**:
  - $n$  é o número total de elementos da amostra.
  - $x_i$  é o número de valores distintos que a variável assume.
  - $F_i$  é a Frequência Absoluta Simples.
  - $f_i$  é a Frequência Relativa Simples.
  - $f_i\%$  é a Frequência Relativa Simples Percentual.  $f_i\% = f_i \cdot 100\%$ .
  - $F_a$  é a Frequência Absoluta Acumulada.

<u><math>x_i</math></u>	<u><math>F_i</math></u>	<u><math>f_i</math></u>	<u><math>f_i\%</math></u>	<u><math>F_a \downarrow</math></u>	<u><math>F_a \uparrow</math></u>	<u><math>f_a \downarrow</math></u>	<u><math>f_a \uparrow</math></u>
0	6	0,2	20	6	30	0,2	1
1	11	0,37	37	17	24	0,57	0,8
2	8	0,27	27	25	13	0,84	0,43
3	2	0,07	7	27	5	0,91	0,16
4	2	0,06	6	29	3	0,97	0,09
6	1	0,03	3	30	1	1	0,03
Total	30	1	100	-	-	-	-

Figure 3: Distribuição tabular quantitativa discreta.

Observação:


As setas simbolizam ordem crescente ou decrescente.

### 5.2.2 Variável Quantitativa Contínua

- Teoria:
  - A construção da representação tabular é realizada de maneira análoga ao caso das variáveis discretas.
  - As frequências são agrupadas em classes, denominadas de “Classes de Frequência”.
  - Denominada “Distribuição de Frequências em Classes” ou “Distribuição em Frequências Agrupadas”.

Dist. Frequências “X ~ Nº de Acidentes por dia, na BR 101, Setembro de 2015”

Nova Representação!



<u><math>x_i</math></u>	<u><math>F_i</math></u>	<u><math>f_i</math></u>	<u><math>f_i\%</math></u>	<u><math>Fa\downarrow</math></u>	<u><math>Fa\uparrow</math></u>	<u><math>fa\downarrow</math></u>	<u><math>fa\uparrow</math></u>
0	6	0,2	20	6	30	0,2	1
1	11	0,37	37	17	24	0,57	0,8
2	8	0,27	27	25	13	0,84	0,43
3	2	0,07	7	27	5	0,91	0,16
4	2	0,06	6	29	3	0,97	0,09
6	1	0,03	3	30	1	1	0,03
Total	30	1	100	-	-	-	-

Fonte: Governo Federal

Figure 4: Distribuição de frequências em classes.

- Convencionar o tipo de intervalo para as classes de frequência:

- Intervalo “exclusive – exclusive”:  $x_i \text{ — } x_j$
- Intervalo “inclusive – exclusive”:  $x_i \text{ — } x_j$
- Intervalo “inclusive – inclusive”:  $x_i \text{ — } x_j$
- Intervalo “exclusive – inclusive”:  $x_i \text{ — } x_j$

OBS.:  $x_i$  - Limite Inferior (LI) de Classe;

$x_j$  - Limite Superior (LS) de Classe;

Figure 5: Intervalo de classes, distribuição de frequências quantitativa continua.

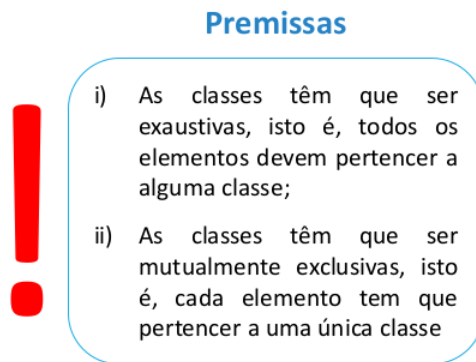


Figure 6: Distribuição frequências quantitativa continua, premissas.



Passos para contruir a **Tabela Distribuição de Frequências Contínua**:

1. Como estabelecer o **número de classes** ( $k$ ):

- Normalmente varia de 5 a 20 classes.
- Critério fórmula de Sturges:

$$k \cong 1 + 3,3 \cdot \log(n)$$

Onde  $n$  é o número de elementos amostrais.

Arredondar  $k$  para número inteiro.

- Critério da Raiz quadrada:

$$k \cong \sqrt{n}$$

Onde  $n$  é o número de elementos amostrais.

Arredondar  $k$  para número inteiro.

2. Como calcular a **Amplitude Total** ( $AT_x$ ):

- Diferença entre o maior e o menor valor observado.
- Intervalo de variação dos valores observados.
- Aproximar valor calculado para múltiplo do nº classes ( $k$ ).
- Garantir inclusão dos valores mínimo e máximo.
- Cálculo:

$$AT_x = Mx(X_i) - Mn(X_i)$$

Onde,

$AT_x$  é a Amplitude Total;

$Mx(X_i)$  é o *valor máximo das amostras*;

$Mn(X_i)$  é o *valor mínimo das amostras*.

- Exemplo:

Se  $k = 5$ ,

$AT_x = 28$

Logo, arredondando  $AT_x = 30$ , para aproximar o valor  $AT_x$  de um múltiplo de  $k$ .

3. Como calcular a **Amplitude das classes da frequência** ( $h$ ):

- As classes terão amplitudes iguais.
- Cálculo:

$$h = h_i = \frac{AT_x}{k}$$

Onde,

$k$  é o **número de classes** e  $AT_x$  é a **Amplitude Total**.

$h$  deve ser arredondado para cima, num número inteiro.

4. Como determinar o ponto médio das classes, representatividade da classe ( $p_i$ ):

$$p_i = \frac{(LS_i - LI_i)}{2}$$

Onde,

$LS_i$  é o limite superior da classe.

$LI_i$  é o limite inferior da classe.

5. Passos da preparação dos dados:

- 1º Passo - **DADOS BRUTOS**:  
Obter os dados da maneira que foram coletados.
- 2º Passo - **ROL**:  
Organizar os DADOS BRUTOS em uma determinada ordem (crescente ou decrescente).
- 3º Passo - **CONSTRUÇÃO TABELA**:  
Na primeira coluna são colocados as classes, e nas demais as respectivas frequências.
- Exemplo:

Nº Classe	Classes (xi)	Fi	fi	fi%	Fa↓	Fa↑	fa↓	fa↑	fa↓%	pi
1	45  --- 52	3	0,08	8	3	40	0,08	1	100	48,5
2	52  --- 59	7	0,18	18	10	37	0,26	0,92	92	55,5
3	59  --- 66	11	0,28	28	21	30	0,53	0,75	75	62,5
4	66  --- 73	10	0,25	25	31	19	0,78	0,47	47	69,5
5	73  --- 80	4	0,10	10	35	9	0,88	0,22	22	76,5
6	80  --- 87	4	0,10	10	39	5	0,98	0,12	12	83,5
7	87  --- 94	1	0,02	2	40	1	1,00	0,02	2	90,5
Total		40	1,00	100	-	-	-	-		-

Fonte: Dados Fictícios

Figure 7: Tabela de distribuição de frequência quantitativa continua.

$X_i$  são as classes.

$F_i$  é a Frequência Absoluta Simples.

$f_i$  é a Frequência Relativa Simples.

$f_i\%$  é a Frequência Relativa Simples Percentual.

$F_a$  é a Frequência Absoluta Acumulada.

$f_a$  é a Frequência Absoluta Acumulada Simples.

$f_a\%$  é a Frequência Absoluta Acumulada Simples Percentual.

$p_i$  é a Representatividade da classe (ponto médio das classes).

### 5.2.3 Variáveis Qualitativas

- Passos da preparação dos dados:
  - Análogo ao procedimento para dados discretos.
  - 1º Passo - **DADOS BRUTOS**:  
Obter os dados da maneira que foram coletados.
  - 2º Passo - **ROL**:  
Nesse caso é feita organização dos DADOS BRUTOS em ordem (Crescente ou Decrescente) de importância.
  - 3º Passo - **CONSTRUÇÃO TABELA** (Com duas ou mais colunas).
- Distribuição de Frequência:
  - $x_i$  é o número de valores distintos que a variável assume.
  - $F_i$  é a Frequência Absoluta Simples.
  - $f_i$  é a Frequência Relativa Simples.
  - $f_i\%$  é a Frequência Relativa Simples Percentual.
  - Inserir comentário sobre os dados.

### 5.3 Medidas de posição

- Localizar a *maior concentração de valores* de uma distribuição.
- *Sintetizar o comportamento* do conjunto do qual ele é originário.
- Possibilitar a *comparação* entre séries de dados.
- As principais **medidas de posição** são:
  - **Média Aritmética** (Simples e Ponderada)
  - **Mediana**
  - **Moda**
  - **Separatrizes**
- Medidas de posição comparação:

## Medidas de Posição - Comparação

Medida	Definição	Vantagens	Desvantagens
Média	Centro da Distribuição	Reflete todos os valores	É afetada por valores extremos
Mediana	Divide a distribuição ao meio	Menos sensível a valores extremos	Difícil determinar para grandes quantidades de dados
Moda	Valor mais frequente	Valor típico	Não é utilizado em análises matemáticas

### 5.3.1 Média Aritmética (Simples e Ponderada)

- **Média Aritmética Simples**, dados Não-Agrupados (não tabelados):
  - **Média Aritmética** ( $\bar{x}$ ) é o valor médio dos dados da distribuição.
  - É a soma de todos os elementos, dividido pelo número total de elementos.
  - Cálculo:

$$\bar{x} = \frac{Soma}{n_{Total}}$$

- **Média Aritmética Ponderada**, dados Agrupados (tabelados):
  - Atribui-se um peso a cada valor da série.
  - É o *Ponto Médio das Classes* ( $p_i$ ), multiplicado por suas respectivas *Frequência Absoluta Simples* ( $F_i$ ), somadas. Dividido pelo *Número Total de Elementos da Amostra* ( $n$ ).
  - Cálculo:

$$\bar{x} = \frac{\sum_{i=1}^n p_i \cdot F_i}{n_{Total}}$$

ou,

$$\bar{x} = \frac{(p_1 \cdot F_1) + (p_2 \cdot F_2) + (p_3 \cdot F_3) + \dots}{n_{Total}}$$

### 5.3.2 Mediana ( $md(x)$ )

#### 5.3.2.1 Mediana Discreta

- Com dados em ROL, é o valor que divide o conjunto de dados em duas partes iguais.
- No caso de número de elementos ímpar, a mediana ( $md(x)$ ) é o elemento central.
- No caso de número de elementos par, a mediana ( $md(x)$ ) é a média aritmética simples dos valores centrais:

$$md(x) = \frac{x_{\frac{n}{2}} + x_{\frac{n+1}{2}}}{2}$$

Onde,

$x$  é a posição do elemento;

$n$  é o número total de elementos.

#### 5.3.2.2 Mediana Contínua

- Mediana ( $md$ ) em distribuição de frequência em variável contínua (dados agrupados em classes):

1. Fazer a coluna da **Frequência Absoluta Acumulada**, que é o somatório das frequências ao longo das classes.
2. Definindo o **Intervalo da Mediana**.

- Obter o número total de elementos  $n$  (somatório das frequências de classes),

$$n = \sum f_i$$

- Determinar a posição do elemento do meio do somatório das frequências:

$$x = \frac{\sum f_i}{2}$$

- A classe que contém essa posição  $x$  na **Frequência Absoluta Acumulada** é a classe do *intervalo da mediana*.

3. Cálculo da Mediana:

$$md = Li + \left( \frac{\frac{\sum f_i}{2} - Fa_{anterior}}{f_{intervalo}} \cdot h \right)$$

Onde,

$Li$  é o limite inferior do *intervalo da mediana*;

$\sum f_i$  é o somatório das frequências (**frequência total** ( $n$ ));

$Fa_{anterior}$  é a **Frequência Absoluta Acumulada** da classe anterior (linha anterior ao *intervalo da mediana*);

$f_{intervalo}$  é a **Frequência Absoluta Simples** do *intervalo da mediana*;

$h$  é a Amplitude da classe do *intervalo da mediana*.

$$h = Ls - Li$$



### 5.3.3 Moda

- Moda ou  $Mo(x)$ : Valor com maior frequência de ocorrência em uma distribuição.
- Podem haver mais de um valor distinto com maior frequência, podendo assim ter mais de um valor na moda.
- Moda com frequência Continua:

1. **Moda Bruta** ( $M_{Bruta}$ ):

- Achar a classe com maior frequência, esse será o *Intervalo Modal*.
- Calcular o *Ponto Médio* (Representatividade da classe) do *Intervalo Modal*:

$$PM = \frac{LS + LI}{2}$$

Onde,

LS = Limite superior da classe;

LI = Limite inferior da classe.

- O *Ponto Médio* do *Intervalo Modal* será a **Moda Bruta** ( $M_{Bruta}$ ).

2. **Moda King** ou **Moda do Rei** ( $M_{King}$ ):

- Determinar o intervalo (classe) com maior frequência, esse será o *Intervalo Modal*.
- Cálculo da Moda de King ( $M_{King}$ ):

$$M_{King} = LI + \left( \frac{F_{post}}{F_{post} + F_{ant}} \cdot h \right)$$

Onde,

$LI$  é o limite inferior da classe do *Intervalo Modal*;

$F_{post}$  é a frequência da classe posterior ao *Intervalo Modal*;

$F_{ant}$  é a frequência da classe anterior ao *Intervalo Modal*;

$h$  é a amplitude do intervalo da classe

$$h = LS - LI$$

3. **Moda de Czuber** ( $M_{Czuber}$ ):

- Determinar o intervalo (classe) com maior frequência, esse será o *Intervalo Modal*.

– Cálculo da **Moda de Czuber** ( $M_{Czuber}$ ):

$$M_{Czuber} = LI + \left( \frac{\Delta_{ant}}{\Delta_{ant} + \Delta_{post}} \cdot h \right)$$

Onde,

$LI$  é o limite inferior da classe do *Intervalo Modal*;

$\Delta_{ant}$  é a variação (diferença) da frequência da classe anterior (ao *Intervalo Modal*) com o *Intervalo Modal* (classe com maior frequência)

$$\Delta_{ant} = |F_i - F_{i-1}|$$

$\Delta_{post}$  é a variação (diferença) da frequência da classe posterior (ao *Intervalo Modal*) com o *Intervalo Modal* (classe com maior frequência)

$$\Delta_{post} = |F_i - F_{i+1}|$$

$h$  é a amplitude do intervalo da classe

$$h = LS - LI$$

### 5.3.4 Separatrizes

- **Separatrizes** são valores da distribuição que a dividem em partes quaisquer.
- A **mediana**, apesar de ser uma medida de tendência central, é também uma **separatriz** de ordem 1/2, ou seja, divide a distribuição em duas partes iguais.
- As **separatrizes** mais comumente usadas são:
  - **Quartis**  
Dividem a distribuição em quatro partes iguais, de ordem 1/4.
  - **Decis**  
Dividem a distribuição em 10 partes iguais, de ordem 1/10.
  - **Centis**  
Dividem a distribuição em 100 partes iguais, de ordem 1/100.
- Fórmula das Separatrizes:

#### 1. Achar o Intervalo da separatriz

- É a classe em que se encontra a separatriz procurada.
- Fazer a coluna de **Frequencia Absoluta Acumulada** ( $F_a$ ).
- É o somatório das frequências (total das frequências), multiplicado pela fração da separatriz procurada ( $k$ ). O resultado é a posição da frequência na coluna **Frequencia Absoluta Acumulada** ( $F_a$ ).

$$P_k = k \cdot \sum f_i$$

A classe na qual a posição pertence é o **Intervalo da separatriz**.

#### 2. Cálculo da separatriz:

$$Sp = L_i + \left( \frac{k \cdot \sum f_i - Fa_{anterior}}{f_{Intervalo}} * h \right)$$

Onde,

$L_i$  é o limite inferior do **Intervalo da separatriz**;

$k$  é a fração (porcentagem) da separatriz procurada;

$\sum f_i$  é o somatório das frequências;

$Fa_{anterior}$  é a **Frequência Absoluta Acumulada** da classe anterior ao **intervalo da separatriz**;

$f_{Intervalo}$  é a **Frequência Absoluta Simples** do **intervalo da separatriz**;

$h$  é a **Amplitude** da classe (limite superior - limite inferior da classe).

$$h = Ls - Li$$

3. Cálculo de **Amplitude Interquartil** ( $AI$ ):

- É a diferença entre 3º quartil e o 1º quartil.

$$AI = Q_3 - Q_1$$

- Para descobrir os valores dos Quartis ( $Q_1$  e  $Q_3$ ) basta usar o *cálculo das separatrizes*.

## 5.4 Medidas de dispersão

- Medem o grau de **variabilidade** (dispersão) dos valores observados em torno da **Média Aritmética**.
- Caracterizam a **representatividade da média** e o nível de **homogeneidade** ou **heterogeneidade** dentro de cada grupo analisado.

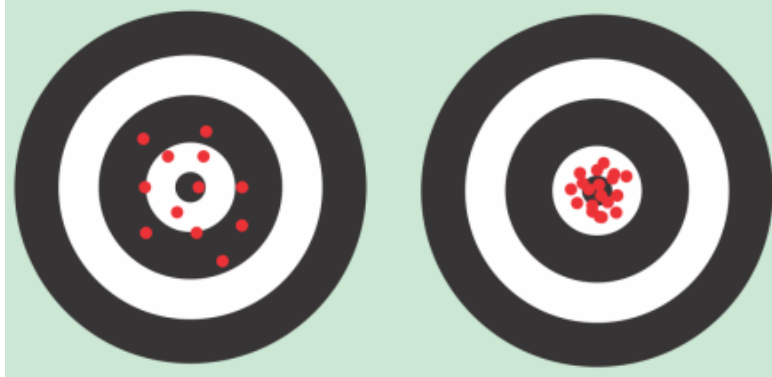


Figure 8: Exemplo de dispersão com heterogeneidade e homogeneidade.

### 5.4.1 Amplitude Total ( $A_T$ )

- Diferença entre o maior e o menor dos valores da série.
- Não considera a dispersão dos valores internos, apenas os extremos.
- Utilização limitada enquanto medida de dispersão, oferece pouca informação.
- Cálculo:

$$A_T = X_{Mx} - X_{Mn}$$

Onde,

$X_{Mx}$  é o valor máximo da série;

$X_{Mn}$  é o valor mínimo da série.

## 5.4.2 Desvio

### 5.4.2.1 Desvio Absoluto ( $D$ )

- Para dados não agrupados:
  - Os **Desvios Absolutos** ( $D$ ) são a diferença absoluta entre um valor observado e a média aritmética:

$$D = |x_i - \bar{X}|$$

Onde,

$x_i$  é o **valor de cada elemento**;

$\bar{x}$  é a **Média Aritmética**.

Os **Desvios Absolutos** ( $D$ ) são um conjunto de elementos como resposta final.

- Para dados agrupados, sem intervalo de classe:

- Cálculo:

$$d_i = |x_i - \bar{X}|$$

Onde,

$x_i$  é o valor da variável discreta;

$\bar{X}$  é a **Média Aritmética**.

- Para dados agrupados, com intervalo de classe:

- Cálculo:

$$d_i = |p_i - \bar{x}|$$

Onde,

$p_i$  é a **Representatividade da classe** (ponto médio da classe);

$\bar{x}$  é a **Média Aritmética Ponderada** calculada para *dados agrupados contínuos*:

$$\bar{x} = \frac{\sum_{i=1}^N p_i \cdot f_i}{\sum f_i}$$

É o *Ponto Médio das Classes* ( $p_i$ ), multiplicado por suas respectivas *Frequência Absoluta Simples* ( $F_i$ ), somadas. Dividido pelo *Número Total de Elementos da Amostra* ( $n$ ).

#### 5.4.2.2 Desvio Absoluto Médio ( $dm$ )

- É a **Média** dos **Desvios**.
- Para dados não agrupados:

– Cálculo:

$$dm(x) = \frac{\sum_{i=1}^n |x_i - \bar{x}|}{n}$$

Onde,

$x_i$  é o **valor de cada elemento**;

$\bar{x}$  é a **Média Aritmética**;

$n$  é o **número total de elementos** (frequencia total).

- Para dados agrupados, sem intervalo de classe:

– Cálculo:

$$D_M = \frac{\sum |d_i| \cdot f_i}{n}$$

Onde,

$d_i$  é o **Desvio Absoluto** para dados agrupados, sem intervalo de classe;

$f_i$  é a **Frequência** de cada variável discreta;

$n$  é o número total de elementos (ou somatório das frequências).

- Para dados agrupados, com intervalo de classe:

– Cálculo:

$$D_M = \frac{\sum |d_i| \cdot f_i}{\sum f_i}$$

Onde,

$d_i$  é o **Desvio Absoluto** para dados agrupados, com intervalo de classe;

$f_i$  é a **frequência** de cada intervalo de classe.

### 5.4.3 Variância ( $\sigma^2$ ou $S^2$ )

- Leva em consideração os valores extremos e também os valores intermediários.
- Relaciona os desvios em torno da média (distancias dos valores ate a média).
- Média Aritmética dos quadrados dos desvios.
- O símbolo para **Variância Populacional** é o sigma ao quadrado ( $\sigma^2$ ), já o símbolo para **Variância Amostral** é o “S” maiusculo ao quadrado ( $S^2$ ).
- Cálculo para dados não agrupados:

– População

$$\sigma^2 = \sum_{i=1}^N \frac{(x_i - \bar{x})^2}{N}$$

Onde,

$x_i$  é o valor de **cada elemento da série**;

$\bar{x}$  é o valor da **Média Aritmética Simples**;

$N$  é o **número total da população**.

– Amostra

$$S^2 = \sum_{i=1}^n \frac{(x_i - \bar{x})^2}{n - 1}$$

Onde,

$x_i$  é o valor de **cada elemento da série**;

$\bar{x}$  é o valor da **Média Aritmética Simples**;

$n$  é o **número de elementos da Amostra**;

$(n - 1)$  é por ser uma estimativa no caso da Amostra, trabalhando assim com um grau a menos de liberdade.

- Cálculo dados agrupados:

– Para dados agrupados, sem intervalo de classe (**Variáveis Discretas**):

\* População

$$\sigma^2 = \frac{\sum (x_i - \bar{X})^2 \cdot f_i}{\sum f_i}$$

Onde,

$x_i$  é o valor de **cada elemento da série**;

$\bar{X}$  é o valor da **Média Aritmética Ponderada**;

$f_i$  é a **Frequência** da variável;

$\sum f_i$  é o somatório das **Frequências**.



\* Amostra

$$S^2 = \frac{\sum (x_i - \bar{X})^2 \cdot f_i}{n - 1}$$

Onde,

$x_i$  é o valor de **cada elemento da série**;

$\bar{X}$  é o valor da **Média Aritmética Ponderada**;

$f_i$  é a **Frequência** da variável;

$n - 1$  ou  $\sum f_i - 1$  é o somatório das **Frequências** da Amostra menos 1.

– Para dados agrupados, com intervalo de classe (**Variáveis Contínuas**):

\* População

$$\sigma^2 = \frac{\sum (p_i - \bar{X})^2 \cdot f_i}{\sum f_i}$$

Onde,

$p_i$  é a **Representatividade das Classe (Ponto Médio das Classes)**;

$\bar{X}$  é o valor da **Média Aritmética Ponderada**;

$f_i$  é a **Frequência** da variável;

$\sum f_i$  é o somatório das **Frequências**.

\* Amostra

$$S^2 = \frac{\sum (p_i - \bar{X})^2 \cdot f_i}{n - 1}$$

Onde,

$p_i$  é a **Representatividade das Classe (Ponto Médio das Classes)**;

$\bar{X}$  é o valor da **Média Aritmética Ponderada**;

$f_i$  é a **Frequência** da variável;

$n - 1$  ou  $\sum f_i - 1$  é o somatório das **Frequências** da Amostra menos 1.

#### 5.4.4 Desvio-padrão ( $\sigma$ ou $S$ )

##### 5.4.4.1 Variância x Desvio-padrão

- **Variância:**
  - Número em unidade “quadrada”.
  - Maior dificuldade de compreensão e menor utilidade na estatística descritiva.
  - Extremamente relevante na inferência estatística e em combinações de amostras.
- **Desvio-padrão:**
  - Mais usado na comparação de diferenças entre conjuntos de dados.
  - Determina a dispersão dos valores em relação a **Média**.
  - Volta-se com os dados para a unidade original.

##### 5.4.4.2 Desvio-padrão (Populacional e Amostral)

- Determina a dispersão dos valores em relação a **Média**.
- População

$$\sigma = \sqrt{\sigma^2}$$

Onde,  
 $\sigma^2$  é a **Variância Populacional**;  
 $\sigma$  é o **Desvio-padrão Populacional**.

- Amostra

$$S = \sqrt{S^2}$$

Onde,  
 $S^2$  é a **Variância Amostral**;  
 $S$  é o **Desvio-padrão Amostral**.

### 5.4.5 Coeficiente de Variação ( $CV$ )

#### 5.4.5.1 Teoria

- Medida relativa de dispersão.
- Útil para comparação em termos relativos do grau de concentração.
- O **Coeficiente de Variação** ( $CV$ ) é expresso em porcentagens.
- Diz-se que uma distribuição:
  - $CV \leq 15\%$  tem **Baixa Dispersão**.
  - $15\% < CV < 30\%$  tem **Média Dispersão**.
  - $CV \geq 30\%$  tem **Alta Dispersão**.

#### 5.4.5.2 Cálculo do Coeficiente de Variação

- População:

$$CV = \frac{\sigma}{\bar{X}} \times 100$$

Onde,

$\sigma$  é o **Desvio-padrão Populacional**;

$\bar{X}$  é a **Média Populacional**.

- Amostra:

$$CV = \frac{S}{\bar{x}} \times 100$$

Onde,

$S$  é o **Desvio-padrão Amostral**;

$\bar{x}$  é a **Média Amostral**.

## 6 Análise Estatística

- Para fazer uma Análise Estatística eficiente de dados, precisamos:
  - Limpar os dados  
Remover os *OUTLIER* (valores atípicos, inconsistentes).
  - Aplicar Estatística Descritiva aos dados  
As medidas de posição (**Média**, **Mediana** e **moda**) e dispersão (**Amplitude Total**, **Desvio**, **Desvio Médio**, **Variância**, **Desvio-padrão** e **Coefficiente de Variação**) são maneiras de descrever os dados.
  - Comparar as medidas dos dados  
Principalmente medidas de dispersão, me especial **Coefficiente de Variação**, são ótimas para comparar dados.
  - Previsão de dados  
A principal técnica é de **Regressão**, porém para aplicar, necessita que os dados estejam limpos e com pouca dispersão (quanto menor, melhor).

## 7 Instruções IF

Instruções IF são testes condicionais.

### 7.1 Testes condicionais

O serne da instrução IF esta uma expressão que deve ser avaliada como **True** ou **False**, chamado teste condicional. Esse teste decide se a instrução deve ser executada.

Teste condicional com resultado **True**, o código dentro do IF será executado.

Teste condicional com resultado **False**, o código dentro do IF não será executado.

### 7.2 Operações lógicas

A Table 12 apresenta as principais operações lógica do python. As operações lógicas retornam **True** ou **False**. A Table 13 mostra exemplos das operações lógicas.

Table 12: Operações Lógicas

Operação	Nome	Função
==	Igual a	Varifica se um valor é igual ao outro.
!=	Diferente de	Varifica se um valor é diferente ao outro.
>	Maior que	Varifica se um valor é maior que outro.
>=	Maior ou igual	Varifica se um valor é maior ou igual a outro.
<	Menor que	Varifica se um valor é menor que outro.
<=	Menor ou igual	Varifica se um valor é menor ou igual a outro.
and	E	Retorna True se ambas as afirmações forem verdadeiras.
or	Ou	Retorna True se uma das afirmações for verdadeiras.
not	Negação	Retorna Falso se o resultado for verdadeiro, ou o contrario.

Table 13: Operações Lógicas Exemplos

Operação	Exemplo
==	$1==1 = \text{True}$
!=	$1!=2 = \text{True}$
>	$5>1 = \text{True}$
>=	$5>=5 = \text{True}$
<	$1<5 = \text{True}$
<=	$1<=4 = \text{True}$
and	$(1==1) \text{ and } (4<5) = \text{True}$
or	$(1==1) \text{ or } (2<1) = \text{True}$
not	$\text{not } (1==1) = \text{False}$

Observações:

- Não confundir = com ==. O sinal de = simples é uma atribuição de valor, enquanto que o sinal == duplo representa “igual a”, sendo um operador lógico.
- Os operadores lógicos de igualdade (== e !=) fazem distinção entre letras maiúsculas e minúsculas.

## 7.3 Testando várias condições

Podemos testar duas (ou mais) condições ao mesmo tempo. Para isso as palavras reservadas `and` e `or` ajudam nesse tipo de situação.

### 7.3.1 Testando várias condições lógicas - AND

O operador lógico `and` nada mais é do que o E da lógica, então podemos comparar duas operações lógicas e compara-las seguindo a ideia da tabela verdade do operador E.

TABELA VERDADE - AND		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure 9: Tabela verdade do operador AND.

Exemplo:

```
age_0 = 22
age_1 = 18
print(age_0 >= 21 and age_1 >= 21)
age_1 = 22
print(age_0 >= 21 and age_1 >= 21)
```

False

True

### 7.3.2 Testando várias condições lógicas - OR

O operador lógico `or` nada mais é do que o OU da lógica, então podemos comparar duas operações lógicas e compara-las seguindo a ideia da tabela verdade do operador OU.

TABELA VERDADE - OR		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure 10: Tabela verdade do operador OR.

Exemplo:

```
age_0 = 22
age_1 = 18
print(age_0 >= 21 or age_1 >= 21)
age_0 = 18
print(age_0 >= 21 or age_1 >= 21)
```

True  
False



## 7.4 Verificando se um valor está em uma lista - IN

Para descobrir se um valor em particular já está em uma lista, utilizamos a palavra reservada `in`.

Exemplo:

```
requested_toppings = ['mushrooms', 'onions', 'pineapple']
print('mushrooms' in requested_toppings)
print('pepperoni' in requested_toppings)
```

True

False

## 7.5 Verificando se um valor não está em uma lista - NOT IN

Para descobrir se um valor em particular não está em uma lista, utilizamos a palavra reservada `not in`.

Exemplo:

```
banned_users = ['andrew', 'carolina', 'david']
user = 'marie'

if user not in banned_users:
    print(user.title() + ", you can post a response if you wish.")
```

Marie, you can post a response if you wish.

## 7.6 Expressões booleanas

Um valor booleano é `True` ou `False`, exatamente como o valor de uma expressão condicional após ter sido avaliada.

Valores booleanos muitas vezes são usados para manter o controle de terminada condição.

```
game_active = True
can_edit = True
```

## 7.7 Instruções IF

Testes condicionais fazem parte das instruções `if`. Há vários tipos de instruções `if`, a escolha depende de quantas condições precisam ser testadas.

Os próximos subtópicos são as possibilidades de instruções `if`.

### 7.7.1 Instruções `if` simples

A instrução `if` mais simples contém um teste e uma ação.

Sintaxe:

```
if teste_condicional:  
    ação
```

Ao avaliar o teste condicional e o resultado for `True`, as ações contidas dentro do `if` são executadas, caso contrário as ações contidas dentro da instrução `if` não são executadas.

Exemplo:

```
age = 19  
if age >= 18:  
    print("You are old enough to vote!")
```

You are old enough to vote!

### 7.7.2 Instruções if-else

Um bloco if-else é semelhante a uma instrução if simples, porém a instrução **else** permite definir ação ou um conjunto de ações executado quando o teste condicional falhar.

Sintaxe:

```
if teste_condicional:
    Ação_True
else:
    Ação_False
```

Exemplo:

```
age = 17
if age >= 18:
    print("You are old enough to vote!")
else:
    print("Sorry, you are too young to vote.")
```

Sorry, you are too young to vote.

### 7.7.3 Sintaxe if-elif-else

Muitas vezes se precisará testar mais de duas situações possíveis, para isso é usado a sintaxe **if-elif-else**. O Python executará apenas um bloco em uma cadeia **if-elif-else**. Cada bloco é executado em sequência, até que algum deles passe. Quando um teste passar, o código após esse teste será executado e o Python ignorará o restante dos testes.

Sintaxe:

```
if teste_condicional_1:
    Ação_teste_1
elif teste_condicional_2:
    Ação_teste_2
else:
    Ação_3
```

Exemplo:

```
age = 12
if age < 4:
    print("You admission cost is $0.")
elif age < 18:
    print("You admission cost is $5.")
else:
    print("You admission cost is $10.")
```

You admission cost is \$5.

#### 7.7.4 Usando vários blocos elif

Podemos usar quantos blocos elif quisermos em nosso código.

Sintaxe:

```
if teste_condicional_1:
    Ação_teste_1
elif teste_condicional_2:
    Ação_teste_2
elif teste_condicional_3:
    Ação_teste_3
elif teste_condicional_4:
    Ação_teste_4
else:
    Ação_5
```

Exemplo:

```
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 5
elif age >= 65:
    price = 5
else:
    price = 10
print("Your admission cost is $" + str(price) + ".")
```

Your admission cost is \$5.

### 7.7.5 Omitindo o bloco else

Python não exige um bloco `else` no final de uma cadeia `if-elif`. As vezes um bloco `else` é útil, outras vezes, é mais claro usar uma instrução `elif` adicional que capture a condição específica de interesse. `else` é uma função que captura tudo. Ela corresponde a qualquer condição não atendida por teste `if` ou `elif` específicos e isso, às vezes, pode incluir dados inválidos ou maliciosos. É uma boa prática considerar usar um último bloco `elif` e omitir o bloco `else`.

Sintaxe:

```
if teste_condicional_1:
    Ação_teste_1
elif teste_condicional_2:
    Ação_teste_2
elif teste_condicional_3:
    Ação_teste_3
elif teste_condicional_4:
    Ação_teste_4
```

Exemplo:

```
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 5
elif age >= 65:
    price = 5
elif age < 65:
    price = 10

print("Your admission cost is $" + str(price) + ".")
```

Your admission cost is \$5.

### 7.7.6 Testando várias condições

A cadeia `if-elif-else` é eficaz, mas é apropriada somente quando você quiser que apenas um teste passe. Assim que encontrar um teste que passe, o interpretador Python ignorará o restante dos testes.

As vezes, porém, é importante verificar todas as condições de interesse. Nesse caso, podemos usar um série de instruções `if` simples, sem blocos `elif` ou `else`.

Em suma, se quiser que apenas um bloco de código seja executado, utilize uma cadeia `if-elif-else`. Se mais de um bloco de código deve ser executado, utilize uma série de instruções `if` independentes.

Sintaxe:

```
if teste_condicional_1:
    Ação_teste_1
if teste_condicional_2:
    Ação_teste_2
if teste_condicional_3:
    Ação_teste_3
if teste_condicional_4:
    Ação_teste_4
```

Exemplo:

```
#Pizzaria
requested_toppings = ['mushrooms','extra cheese']

if 'mushrooms' in requested_toppings:
    print("Adding mushrooms.")
if 'pepperoni' in requested_toppings:
    print("Adding pepperoni.")
if 'extra cheese' in requested_toppings:
    print("Adding extra cheese.")

print("\nFinished making your pizza!")
```

Adding mushrooms.

Adding extra cheese.

Finished making your pizza!

**7.8 Usando instruções `if` com listas**

**7.9 Verificando itens especiais**

**7.10 Verificando se uma lista não está vazia**

**7.11 Usando várias listas**

## Bibliografia

- [1] Eric Matthes. *Curso Intensivo de Python - 3ª Edição: Uma Introdução Prática e Baseada em Projetos à Programação*. Novatec Editora, 2023. ISBN: 9788575228432. URL: <https://books.google.com.br/books?id=mkW7EAAAQBAJ>.