



escola  
britânica de  
artes criativas  
& tecnologia

# Desenvolvedor Full Stack Python

Variáveis e Tipos de Dados

# Módulo 01 | Python: Variáveis & Tipos de Dados

Caderno de Aula

Professor [André Perez](#)


---

## Tópicos

1. Introdução ao Google Colab;
  2. Variáveis;
  3. Números;
  4. *Strings*;
  5. Boleanos.
- 

## Aulas

### 1. Introdução ao Google Colab

 Ferramenta web autogerenciada de cadernos (*notebooks*).

#### 1.1. Ferramenta web

- Crie uma conta Google em [gmail.com](https://gmail.com);
- Acesse o Google Colab através do endereço [colab.research.google.com](https://colab.research.google.com).

#### 1.2. Autogerenciada

- A Google provisiona uma máquina virtual para você;
- A máquina virtual dura no máximo 12h.

#### 1.3. Cadernos (*notebooks*)

Um **caderno** é um documento *web* composto por um conjunto de elementos (células) de texto

e código:

- Células de **texto** podem ser editados com o editor da ferramenta, HTML ou Markdown;
- Células de **código** são exclusivamente para a linguagem de programação Python.

```
In [ ]: print("olá mundo!")
```

## 2. Variáveis

### 2.1. Definição

Mecanismo de armazenamento volátil de dados, ou seja, dados salvos em pedacinhos da memória RAM do sistema computacional em uso (*notebook*, *mobile*, console de vídeo game, *smartwatch*, etc.).

```
In [ ]: idade = 30
        print(idade)

        idade = 27
        print(idade)

        nome = "andré"
        print(nome)
```

### 2.2. Tipos nativos

**Tipos numéricos:** inteiros ( `int` ) e decimais ( `float` ):

```
In [ ]: preco = 1000
        tipo_preco = type(preco)

        print(preco)
        print(tipo_preco)

        juros = 0.05
        tipo_juros = type(juros)

        print(juros)
        print(tipo_juros)
```

**Tipos de texto:** *strings* ( `str` ):

```
In [ ]: primeiro_nome = "André"

        print(primeiro_nome)
        print(type(primeiro_nome))

        pais = 'Brasil'

        print(pais)
        print(type(pais))
```

**Tipos lógicos:** booleanos ( `bool` ):

```
In [ ]: usuario_maior_de_idade = True
```

```
print(usuario_maior_de_idade)
print(type(usuario_maior_de_idade))
```

Tipo vazio ( `NoneType` ):

```
In [ ]: telefone_fixo = None

print(telefone_fixo)
print(type(telefone_fixo))
```

## 3. Números

### 3.1. Motivação

Você precisa calcular o **ticket médio** diário `tk` do seu restaurante. A métrica é calculada pela soma do valor das vendas `sv` de um mesmo dia dividido pela quantidade de vendas `sq`, também de um mesmo dia.

$$\text{\$tk} = \text{sv} / \text{sq}$$

Esta é a sua planilha:

Dia	Valor Total Vendas	Qtd Total Vendas	Ticket Medio
19/01	153.98	3	?
20/01	337.01	7	?
23/01	295.33	5	?

Como podemos fazer este cálculo usando o Python?

### 3.2. Definição

Armazenam **valores numéricos**:

- `10, 37, 500` (inteiros);
- `0.333, 10.1` (decimais);
- `1 + 2j` (complexos).

São dos tipos:

- `int` (inteiros);
- `float` (decimais);
- `complex` (complexos).

```
In [ ]: print(type(37))
print(type(10.1))
print(type(1 + 2j))
```

### 3.3. Operações

As operações dos tipos numéricos são as quatro operações matemáticas fundamentais:

- `+` (soma);
- `-` (subtração);
- `*` (multiplicação);

- `/` (divisão).

Além de operações mais avançadas:

- `//` (divisão inteira)
- `**` (potência ou exponenciação);
- `%` (resto de divisão).

**Exemplo:** Carrinho de compra de um *e-commerce*.

```
In [ ]: qtd_items_carrinho_compra = 0

qtd_items_carrinho_compra = qtd_items_carrinho_compra + 1
print(qtd_items_carrinho_compra)

qtd_items_carrinho_compra = qtd_items_carrinho_compra + 1
print(qtd_items_carrinho_compra)
```

```
In [ ]: qtd_items_carrinho_compra = 0

qtd_items_carrinho_compra += 1
print(qtd_items_carrinho_compra)

qtd_items_carrinho_compra += 1
print(qtd_items_carrinho_compra)
```

**Exemplo:** Total a pagar de um produto.

```
In [ ]: preco = 47
quantidade = 0.250

total_a_pagar = quantidade * preco
print(total_a_pagar)
```

```
In [ ]: a = 3
b = 2

c = a / b
print(c)
print(type(c))

d = a // b
print(d)
print(type(d))
```

### 3.4. Conversão

Podemos converter os tipos numéricos entre si utilizando o método nativo `int`, `float` e `complex`:

```
In [ ]: print(int(3.9))
```

```
In [ ]: print(float(10))
```

```
In [ ]: print(complex(1))
```

## 3.5. Revisitando a motivação

Dia	Valor Total Vendas	Qtd Total Vendas	Ticket Medio
19/01	153.98	3	?
20/01	337.01	7	?
23/01	295.33	5	?

Ticket médio diário do dia 19/01.

```
In [ ]: svv_19 = 153.98
        sqv_19 = 3

        tkt_19 = svv_19 / sqv_19
        print(tkt_19)
```

Ticket médio diário do dia 20/01.

```
In [ ]: svv_20 = 337.01
        sqv_20 = 7

        tkt_20 = svv_20 / sqv_20
        print(tkt_20)
```

Ticket médio diário do dia 23/01.

```
In [ ]: svv_23 = 295.33
        sqv_23 = 5

        tkt_23 = svv_23 / sqv_23
        print(tkt_23)
```

Ticket médio

```
In [ ]: tkt = (tkt_19 + tkt_20 + tkt_23) / 3
        print(tkt)
```

## 4. Strings

### 4.1. Motivação

A empresa que você trabalha adquiriu uma *startup* de logística. Você precisa identificar todos endereços que são comum a ambas. Na sua empresa, você armazena a latitude e longitude dos endereços em duas variáveis `lat` e `lon`, já a *startup* adquirida em uma única variável `latlon`.

```
In [ ]: # sua empresa
        lat = '-22.005320'
        lon = '-47.891040'

        # startup adquirida
        latlon = '-22.005320;-47.891040'
```

Como podemos normalizar a forma com que as latitudes e longitudes são armazenadas para possam ser comparadas?

## 4.2. Definição

Armazenam **textos**:

- `c`, `EBAC`, `Andre Perez`, `20 anos` (texto)

São do tipo `str`:

```
In [ ]: nome_aula = 'Aula 04, Módulo 01, Strings'

print(nome_aula)
print(type(nome_aula))
```

```
In [ ]: string_vazia = ""

print(string_vazia)
print(type(string_vazia))
```

## 4.3. Operações

As operações de variáveis do tipo *string* são:

- `+` (concatenação).

**Exemplo:** Nome completo.

```
In [ ]: nome = 'Andre Marcos'
sobrenome = 'Perez'

apresentacao = 'Olá, meu nome é ' + nome + ' ' + sobrenome + '.'
print(apresentacao)
```

Uma outra forma de concatenar strings é utilizar operações de formatação:

```
In [ ]: nome = 'Andre Marcos'
sobrenome = 'Perez'

apresentacao = f'Olá, meu nome é {nome} {sobrenome}.'
print(apresentacao)
```

Outra operação muito utilizada é a de fatiamento (*slicing*):

**Exemplo:** Informações de email.

```
In [ ]: email = 'andre.perez@gmail.com'
```

Fatiamento fixo:

a	n	d	r	e	.	p	e	r	e	z	@	g	m	a	i	l	.	c	o	m
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

```
In [ ]: print('0: ' + email[0])
```

```
print('11: ' + email[11])
```

```
In [ ]: print('-1: ' + email[-1])
        print('-2: ' + email[-2])
```

Fatiamento por intervalo:

	a	n	d	r	e	.	p	e	r	e	z	@	g	m	a	i	l	.	c	o	m
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

```
In [ ]: email_usuario = email[0:11]
        print(email_usuario)
```

```
In [ ]: email_provedor = email[12:21]
        print(email_provedor)
```

## 4.4. Métodos

São métodos nativos do Python que nos ajudam a trabalhar no dia a dia com *strings*.

```
In [ ]: endereco = 'Avenida Paulista, 1811, São Paulo, São Paulo, Brasil.'
```

```
In [ ]: # maiusculo: string.upper()
        print(endereco.upper())
```

```
In [ ]: # posicao: string.find(substring)
        posicao = endereco.find('Brasil')
        print(posicao)
```

```
In [ ]: # substituição: string.replace(antigo, novo)
        print(endereco.replace('Avenida', 'Av'))
```

## 4.5. Conversão

Podemos converter *strings* em tipos numéricos e vice-versa.

```
In [ ]: idade = 19
        print(type(idade))

        idade = str(idade)
        print(type(idade))
```

```
In [ ]: faturamento = 'R$ 35 mi'
        print(faturamento)
        print(type(idade))

        faturamento = int(faturamento[3:5])
        print(faturamento)
        print(type(faturamento))
```



## 4.6. Revisitando a motivação

Encontrando a posição do caracter `;` de divisão das *strings* de latitude e longitude da variável da startup:

```
In [ ]: posicao_char_divisao = latlon.find(';')
        print(posicao_char_divisao)
```

Extraindo a latitude:

```
In [ ]: lat_startup = latlon[0:posicao_char_divisao]
        print(lat_startup)
```

Extraindo a longitude:

```
In [ ]: lon_startup = latlon[posicao_char_divisao+1:len(latlon)]
        print(lon_startup)
```

## 5. Booleans

### 5.1. Motivação

Em *websites* (redes sociais, *e-commerce*, corporativos, etc.) é comum o uso de sistemas de controle de acesso, o famoso *login*. Em geral, nestes sistemas um usuário fornece dois dados:

`usuario` e `senha`:

```
In [ ]: usuario = 'andre.perez'
        senha = 'andre123'
```

Do lado do servidor, o *backend* do *website* tem armazenado os dados de usuário e senha fornecidas pelo usuário no momento do cadastro: `usuario_cadastro` e

`senha_cadastro`:

```
In [ ]: usuario_cadastro = 'andre.perez'
        senha_cadastro = 'andre321'
```

Como comparamos se as *strings* (`usuario`, `usuario_cadastro`) e (`senha`, `senha_cadastro`) são iguais para conceder ou bloquear o acesso do usuário?

### 5.2. Definição

Armazenam **valores lógicos**:

- `True` (verdadeiro);
- `False` (falso).

```
In [ ]: verdadeiro = True
        print(verdadeiro)
```

```
In [ ]: falso = False
        print(falso)
```

São do tipo `bool`.

```
In [ ]: print(type(True))
```

São resultados de comparações lógicas. Os operadores de comparação lógica são:

- `>` (maior);
- `<` (menor);
- `==` (igual);
- `>=` (maior ou igual);
- `<=` (menor ou igual);
- `!=` (diferente).

**Exemplo:** Caixa eletrônico

```
In [ ]: saldo_em_conta = 200
valor_do_saque = 100

pode_executar_saque = valor_do_saque <= saldo_em_conta
print(pode_executar_saque)
```

**Exemplo:** Cartão de crédito

```
In [ ]: codigo_de_seguranca = '852'
codigo_de_seguranca_cadastro = '010'

pode_efetuar_pagamento = codigo_de_seguranca == codigo_de_seguranca_cadastro
print(pode_efetuar_pagamento)
```

### 5.3. Operações

As operações de variáveis booleanas são:

- `|` (operador ou)
- `&` (operador e)
- `not` (operador não)

O conjunto de resultados de operações lógicas geralmente é resumido em uma tabela chamada "tabela da verdade":

A	B	A OR B	A AND B	NOT A
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE

**Exemplo:** Tabela da verdade do operador `|` (ou).

```
In [ ]: print(True | True)
print(True | False)
print(False | False)
print(False | True)
```

**Exemplo:** Tabela da verdade do operador `&` (e).

```
In [ ]: print(True & True)
        print(True & False)
        print(False & False)
        print(False & True)
```

**Exemplo:** Tabela da verdade do operador `not` (não).

```
In [ ]: print(not True)
        print(not False)
```

## 5.4. Conversão

Podemos converter tipos numéricos e *strings* para booleanos através do método nativo `bool` :

```
In [ ]: idade = 19
        tipo_sangue = 'O-'
        filhos = 0
        telefone_fixo = None
        telefone_fixo = ''

        print(bool(idade))
        print(bool(tipo_sangue))
        print(bool(filhos))
        print(bool(telefone_fixo))
        print(bool(telefone_fixo))
```

## 5.5. Revisitando a motivação

Compara se os dados fornecidos pelo usuário são iguais aos dados do cadastro:

```
In [ ]: usuario_igual = usuario == usuario_cadastro
        senha_igual = senha == senha_cadastro

        print(usuario_igual)
        print(senha_igual)
```

Decide se concede o acesso:

```
In [ ]: conceder_acesso = usuario_igual & senha_igual
        print(conceder_acesso)
```

---