

# Introdução à Linguagem Python



**Conteudista:** Prof. Me. Hugo Batista Fernandes

**Revisão Textual:** Esp. Jéssica Dante

## Objetivos da Unidade:

- Explorar primeiros passos no uso do *Python* para criação de programas de computadores;
- Apresentar as principais características da linguagem *Python*, como configurar o ambiente de desenvolvimento, conceitos de variáveis e tipos de dados.



 Material Teórico



 Material Complementar



 Referências



# Material Teórico

---

## Introdução à Linguagem Python

*Python* é uma linguagem de programação de alto nível interpretada, interativa, orientada a objetos. Foi criado por Guido van Rossum no final da década de 1980. Como o Perl, o código-fonte do *Python* também está disponível sob a GNU *General Public License* (GPL). *Python* tem seu nome inspirado um programa de TV inglês chamado “*Flying Circus de Monty Python*” e não na espécie de cobra.

*Python* é uma linguagem de programação de aplicação geral e possui uma sintaxe simples e fácil de usar. Isso torna o *Python* uma excelente linguagem para aprender a programar, destacamos algumas características:

- **Interpretado:** *Python* é processado em tempo de execução pelo interpretador. Não é preciso compilar o programa antes de executá-lo;
- **Orientado a objetos:** *Python* suporta estilo ou técnica de programação orientada a objetos que encapsula código dentro de objetos;
- **Uso geral:** *Python* pode ser usado para quase tudo. É aplicável a quase todos os campos para uma variedade de tarefas. Seja a execução de tarefas de curto prazo como teste de *software* ou desenvolvimento de *software* para uso ao longo prazo;
- **Fácil de aprender:** *Python* é extremamente fácil de começar. *Python* possui uma sintaxe muito simples;

- **Livre e de código aberto:** *Python* é um exemplo de *Software* Livre e Código Aberto. Em linhas gerais, você pode distribuir gratuitamente cópias deste *software*, ler seu código-fonte, fazer alterações nele, usar partes dele em novos programas gratuitos. Seu uso é totalmente gratuito, mesmo para fins comerciais;
- **Portátil:** Devido à sua natureza de código aberto, programas *Python* podem funcionar em diversas plataformas;
- **Case-sensitive:** *Python* diferencia letras maiúsculas de minúsculas em sua codificação;
- **Linguagem com tipagem dinâmica (*Duck typing*):** Em *Python*, não é preciso declarar o tipo de dados ao declarar uma variável. O interpretador determina o tipo de dados em tempo de execução.

---

## Leitura

### *Duck Typing com Python*

Clique no botão para conferir o conteúdo.

ACESSE

Como exemplos de aplicações desenvolvidas com *Python*, podemos destacar:

- **Desenvolvimento Web:** *Python* oferece diferentes *Frameworks* para desenvolvimento *web* como *Django*, *Pyramid*, *Flask*. Esses *Frameworks* são conhecidos pela segurança, flexibilidade e escalabilidade;
- **Desenvolvimento de jogos:** *PySoy* e *PyGame* são duas bibliotecas *Python* usadas para desenvolvimento de jogos;
- **Inteligência Artificial e Aprendizado de Máquina:** Há um grande número de bibliotecas de código aberto que podem ser usadas durante o desenvolvimento de aplicações de Inteligência Artificial e Aprendizado de Máquina;
- **Desenvolvimento de aplicações para Desktop:** O *Python* possui diversos *Frameworks* com as quais podemos construir aplicativos de área de trabalho. *PyQt*, *PyGtk*, *PyGUI* são alguns exemplos.

---

## Leitura

*Framework* de Desenvolvimento: quais os mais usados?

Clique no botão para conferir o conteúdo.

ACESSE

Nas próximas etapas de nossa Unidade de estudo, iremos aprender como baixar o *Python*, criar nossos primeiros códigos e estudar sobre conceitos de variáveis. Ao longo dos estudos, será proveitoso visitar em paralelo, conceitos de desenvolvimento de algoritmos.

## Instalando o Ambiente de Desenvolvimento

Nessa etapa, iremos instalar o *Python* em um ambiente com sistema operacional *Windows*. Serão instalados bibliotecas padrão do *Python*, compilador, interpretador e o editor de *scripts* IDLE.

1

Faça o *download* do instalador do *Python* 3 com a versão mais recente por meio do *site*;

---

**Site**

***Download Python***

**Clique no botão para conferir o conteúdo.**

ACESSE



## Figura 1 – Página de *download* do Python

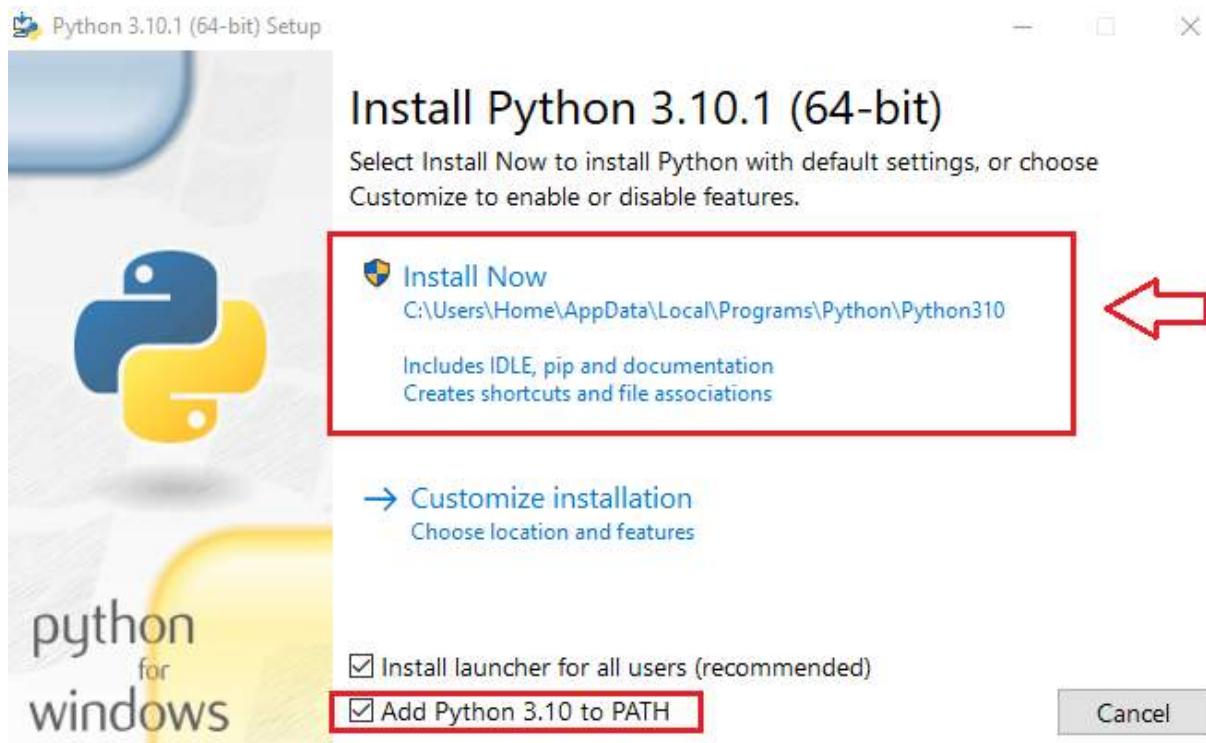
Fonte: Adaptada python.org

2

Localize o arquivo de instalação e o execute. Na tela inicial, na parte inferior, selecione a opção “Add Python 3.10 to PATH”, essa opção fará com que o instalador crie as variáveis de sistema no *Windows*;

3

Em seguida, clique em “*Install Now*”;



**Figura 2 – Software de instalação do Python**

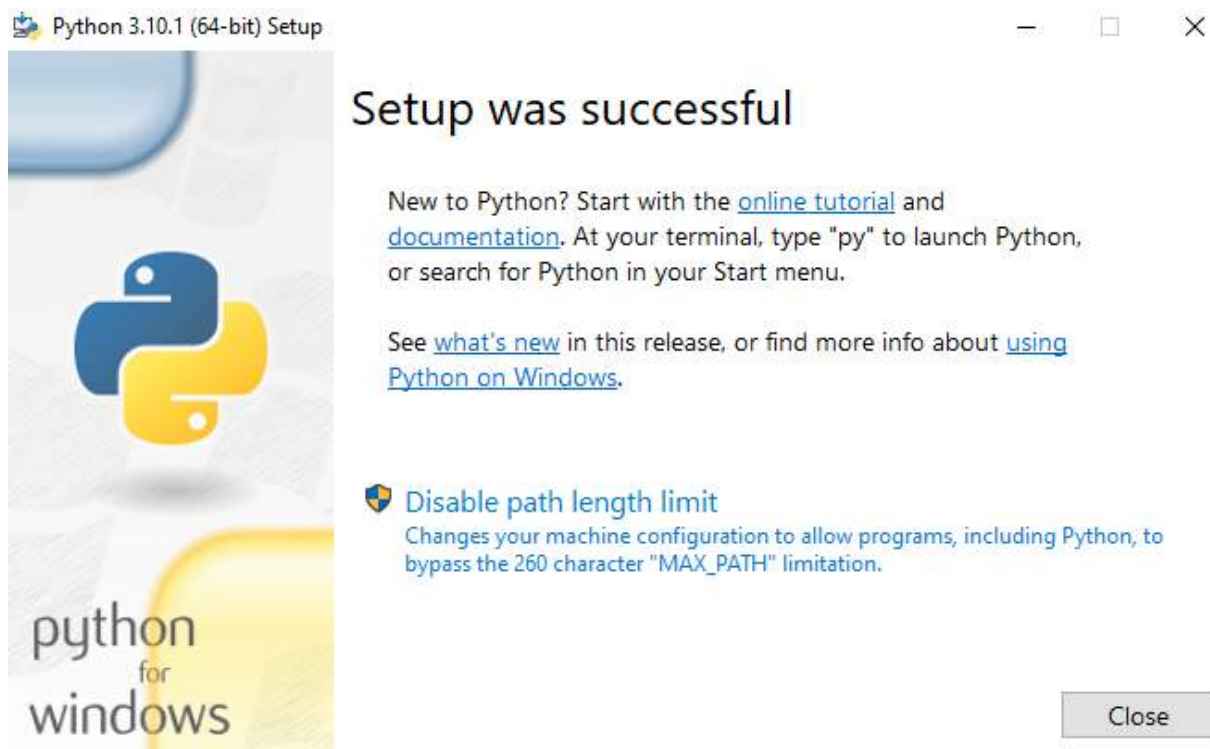
Fonte: Reprodução

4

Caso seu Windows solicite permissão para seguir a instalação, clique em SIM;

5

Aguarde o final da instalação. Na tela a seguir, clique no botão “Close”.



**Figura 3 – Software de instalação do Python**

Fonte: Reprodução

---

## Sites

**Caso não consiga instalar o *Python*, os estudos poderão seguir utilizando ferramentas *online* de compilação. Como sugestão, pode-se utilizar as ferramentas indicadas a seguir.**

***Jdoodle – Online Python 3 IDE***

Clique no botão para conferir o conteúdo.



ACESSE

**Replit – Python (with Turtle) Online Compiler & Interpreter**

Clique no botão para conferir o conteúdo.

ACESSE

---

## Criando Programas em *Python*

Na etapa anterior, instalamos o *Python* em nosso sistema operacional e junto com ele, foi instalado também a ferramenta para desenvolvimento de programas em *Python*, o IDLE, um editor de *scripts* que iremos utilizar a partir de agora para executar e criar nossos programas em *Python*.

---

## Leitura

### Modo Interativo – *Python*

Clique no botão para conferir o conteúdo.

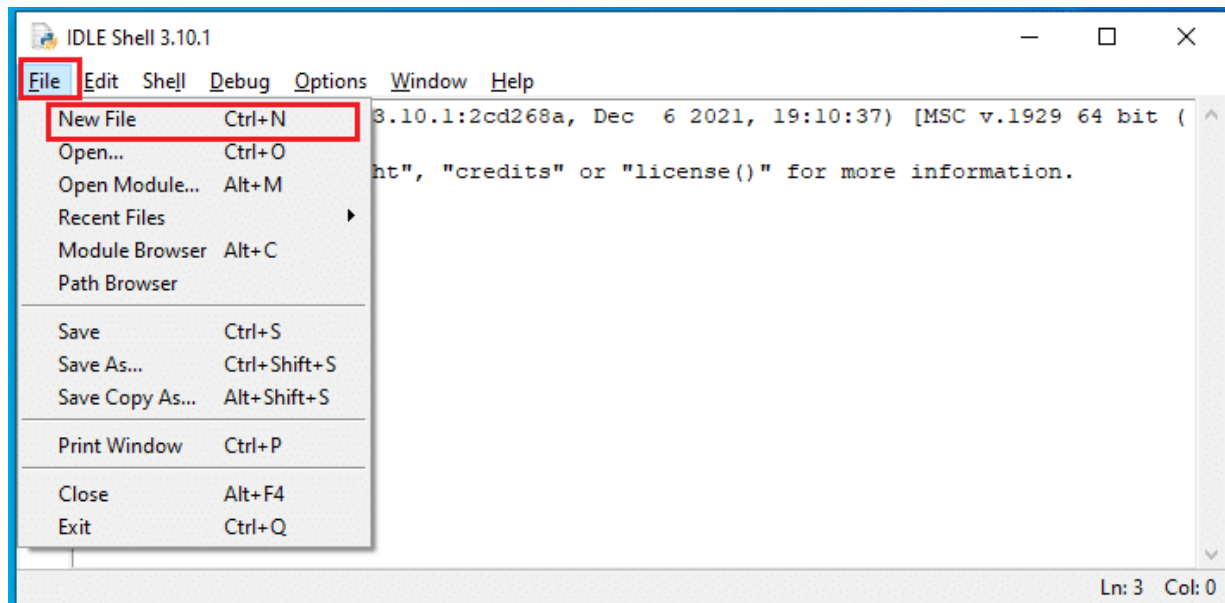
ACESSE

Nessa etapa, iremos aprender como criar e executar nossos códigos em *Python*.

Para criar códigos em *Python*, iremos utilizar o IDLE, para isso, localize-o em seu sistema operacional e execute o programa.

Ao iniciar o IDLE, a tela aberta será o interpretador *Python*. Essa aplicação será a responsável por executar nossos programas em *Python*.

Para criar um programa, clique no botão “File” e em seguida “New File”.



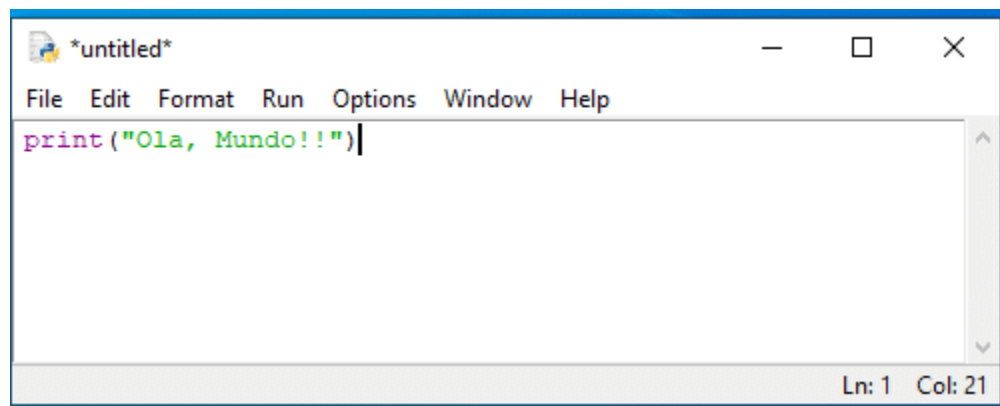
**Figura 4 – Tela do interpretador *Python***

Fonte: Acervo do Conteudista

Para o nosso primeiro programa, iremos criar o tradicional “olá, mundo”. Para isso, usaremos a função “*print*” do *Python*. Utilizamos essa função para apresentar alguma informação na tela, para isso, entre os parênteses da função *print* e entre aspas duplas,

digitamos o que queremos que seja visualizado na tela. Assim, digite na tela de digitação de *scripts*:

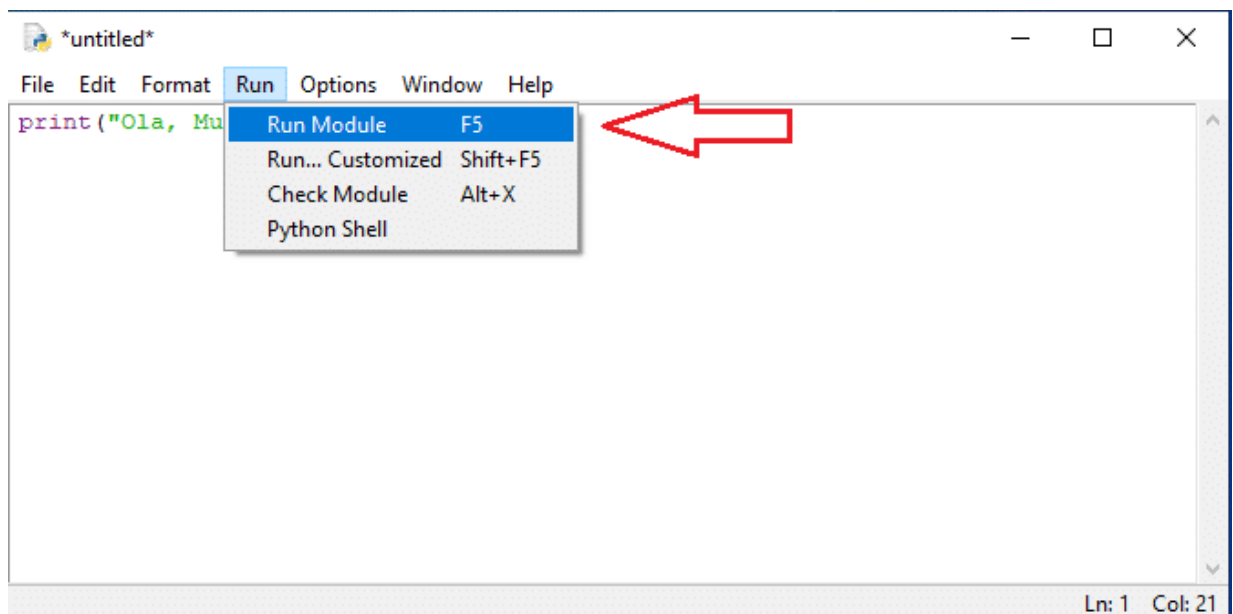
```
print("Ola, Mundo!!")
```



**Figura 5 – Tela de digitação de *scripts***

Fonte: Acervo do Conteudista

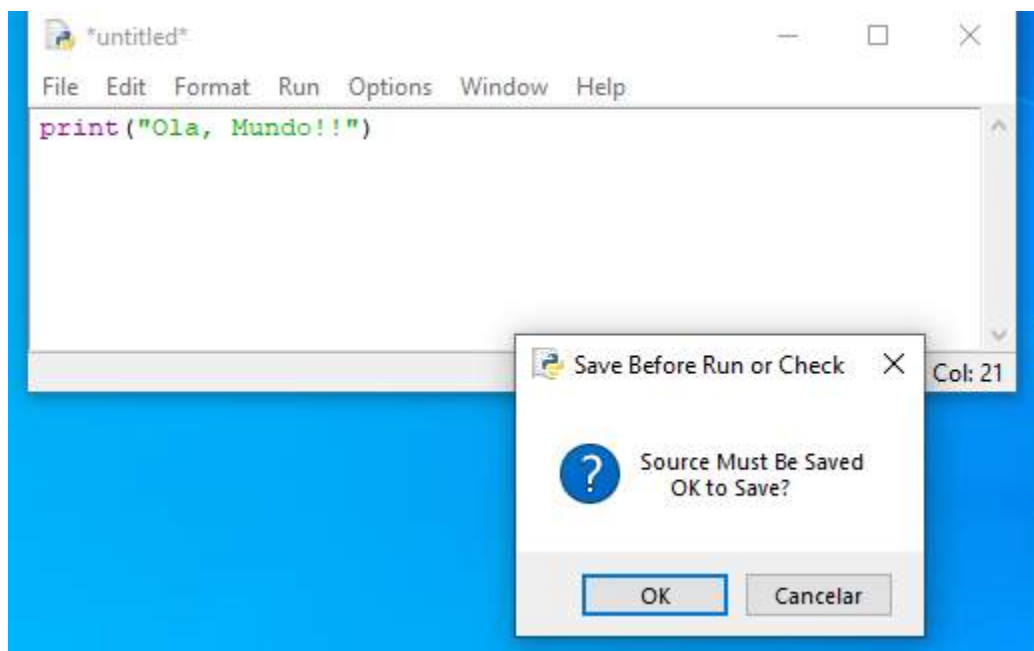
Em seguida, para executar o programa, clique no botão “Run”, opção *Run Module*. Você pode também pressionar a tecla F5 de seu teclado.



**Figura 6 – Tela de digitação de *scripts* – Executando *scripts***

Fonte: Acervo do Conteudista

Como o arquivo não foi salvo, a ferramenta solicita que se salve o programa. Clique no botão “OK” e em seguida escolha um local em seu computador e salve seu programa.

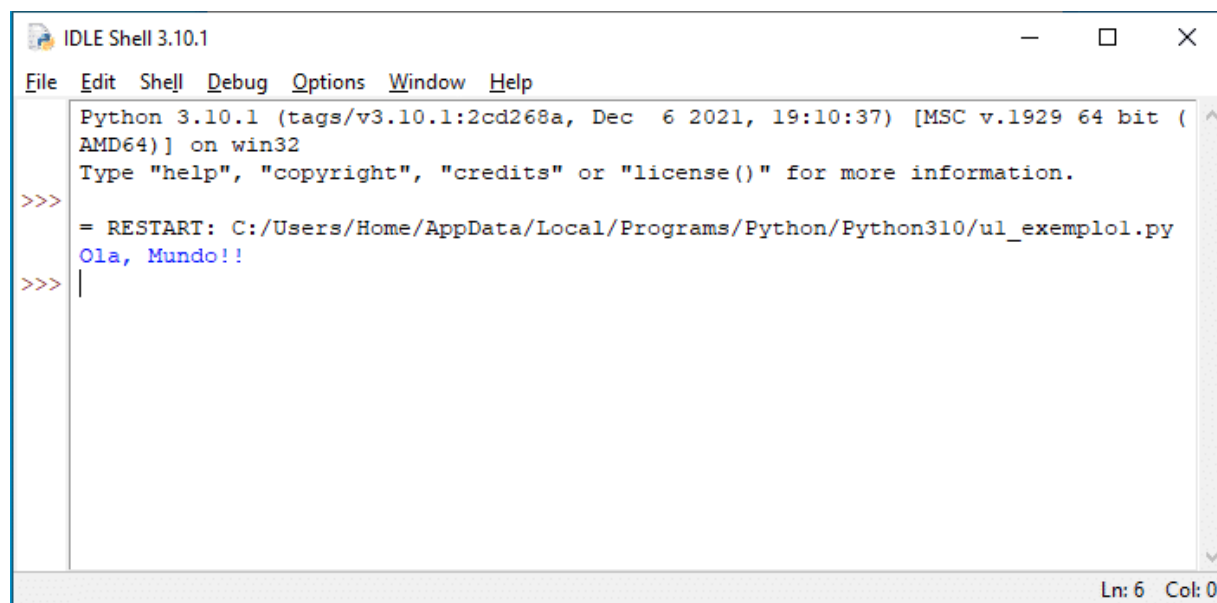


## Figura 7 – Tela de digitação de *scripts* – Salvando *scripts*

Fonte: Acervo do Conteudista

---

Ao concluir, o programa será executado por meio do interpretador *Python* do IDLE.



## Figura 8 – Tela do interpretador *Python*

Fonte: Acervo do Conteudista

---

Agora em diante, qualquer alteração no programa, ao executar, o programa o interpretador *Python* do IDLE será aberto.

## Conceito e Uso de Variáveis no *Python*

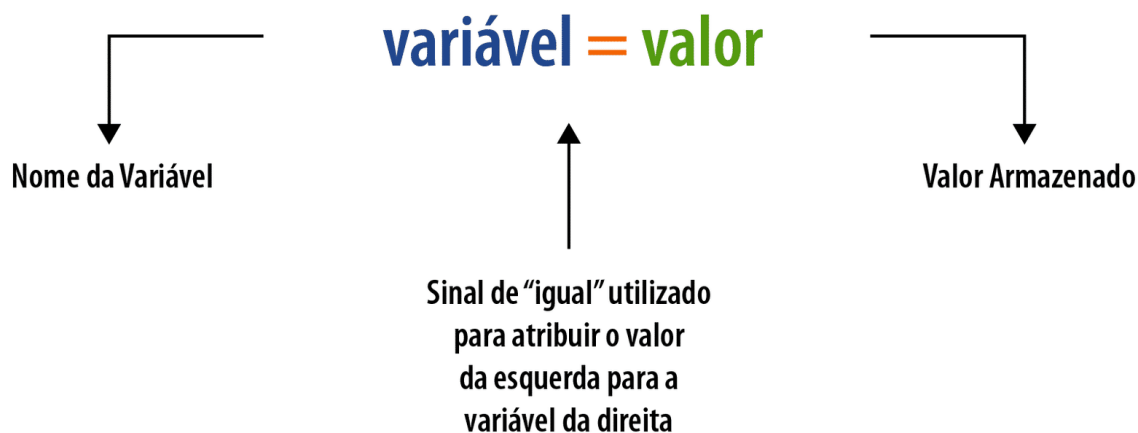
Inicialmente, para seguir com os estudos de conceitos de variáveis, precisamos entender o conceito de **identificadores**. Identificadores em *Python* são os nomes utilizados para identificar variáveis, funções, classes, módulos ou outros objetos.

Vale lembrar que a linguagem *Python* é *Case-sensitive*, ou seja, diferencia letras maiúsculas de minúsculas em sua codificação. Assim, por exemplo, o identificador “pessoa” é diferente de “Pessoa”.

Afinal, o que são variáveis? Uma variável é uma área de armazenamento que nossos programas podem manipular, são localizações de memória reservadas para armazenar valores, desse modo, ao criar uma variável, é reservado algum espaço na memória do computador que está executando o programa.

Por conta da característica de tipagem dinâmica, ao declarar uma variável, não é preciso indicar de forma explícita o tipo de dados que serão armazenados. A declaração ocorre automaticamente quando você atribui um valor a uma variável.

O sinal de igual (=) é usado para atribuir valores às variáveis. A descrição à esquerda do operador = é o nome da variável e a descrição à direita do operador = é o valor armazenado na variável.



**Figura 9**

Para nomear variáveis no *Python*, devemos seguir algumas regras:

- As variáveis podem ter letras (A-Z e a-z), dígitos (0-9) e sublinhados (\_);
- Não podem começar com números;
- Não podem conter pontuação, caracteres especiais (!, @, \$, # etc.) ou espaços;
- Não podem ser utilizadas palavras reservadas da linguagem.

A lista a seguir mostra as palavras-chave *Python*. Estas são palavras reservadas e você não pode usá-las como constantes, variáveis ou qualquer outro nome de identificador. Todas as palavras-chave *Python* contêm apenas letras minúsculas.

**Tabela 1**

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

Os tipos de dados armazenados em uma variável podem ser:

- Números (inteiros, reais e complexos);
- *String* (cadeia de caracteres que representam um texto);
- *Boolean*;
- *List*;
- *Tuple*;
- *Dicionary*.

## Exemplos Práticos: Atribuindo e Manipulando Valores para Variáveis

Em nosso primeiro exemplo, vamos criar um programa que seja capaz de armazenar o nome de uma pessoa, sua idade e *e-mail*. Nosso programa irá exibir na tela os valores armazenados.

Digite o seguinte código no editor de *scripts* do IDLE:

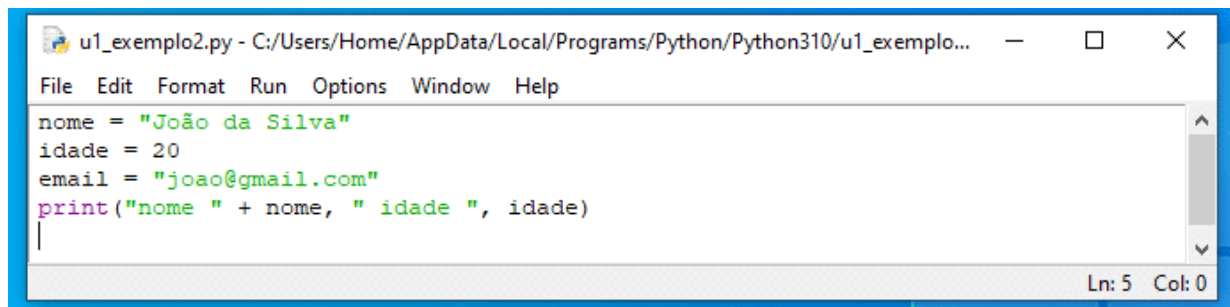
```
nome = "João da Silva"

idade = 20

email = "joao@gmail.com"
```



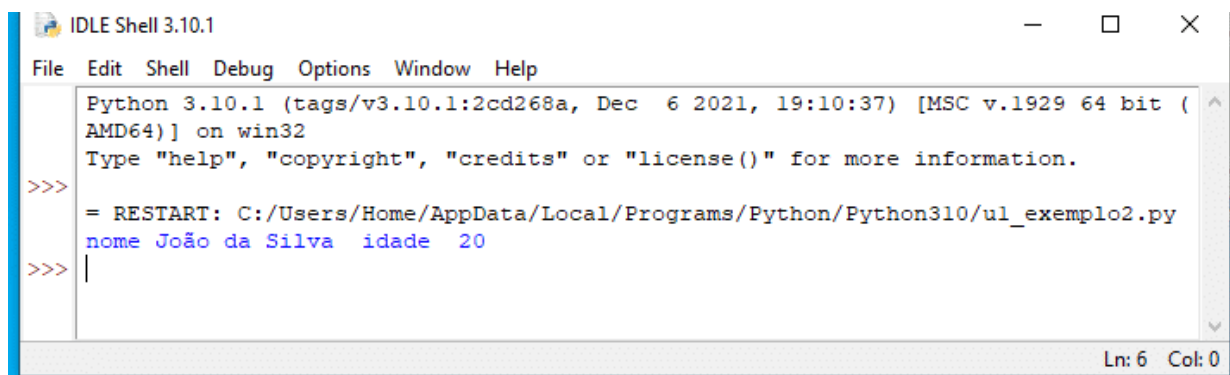
```
print("nome " + nome, " idade ", idade)
```



**Figura 10**

Fonte: Acervo do Conteudista

Saída:



**Figura 11**

Fonte: Acervo do Conteudista

## Explicando o Código

- **Na linha 1:** atribuímos o valor “João da Silva” à variável **nome**. Quando desejamos armazenar um valor de sequência de caracteres (*string*), devemos digitar o valor entre aspas duplas;
- **Na linha 2:** Atribuímos o valor “20” à variável **idade**. Diferentemente da atribuição anterior, por se tratar de um número, não precisamos digitar o número entre aspas duplas. Quando desejarmos armazenar um valor, precisamos apenas digitar os números (utilize ponto ao invés de vírgula para separar casas decimais);
- **Na linha 3:** Atribuímos o valor “joao@gmail.com” a variável **email**;
- **Na linha 4:** Por meio da função “*print*”, exibimos os valores armazenados nas variáveis (impresso na tela). Concatenamos os valores das variáveis e rótulos.

---

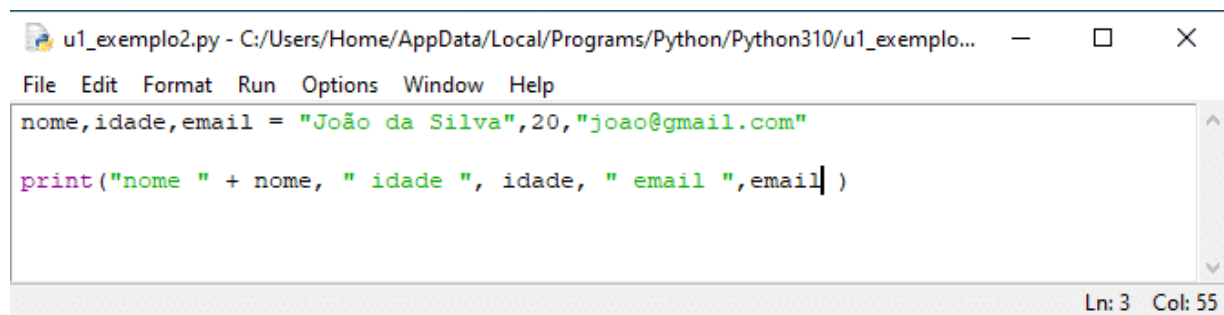
## Leitura

### *Python função print()*

Clique no botão para conferir o conteúdo.

ACESSE

Podemos atribuir valores em uma mesma linha a múltiplas variáveis. Para isso, basta digitar os valores separados por vírgula.



```
u1_exemplo2.py - C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo...
File Edit Format Run Options Window Help
nome,idade,email = "João da Silva",20,"joao@gmail.com"

print("nome " + nome, " idade ", idade, " email ",email )
```

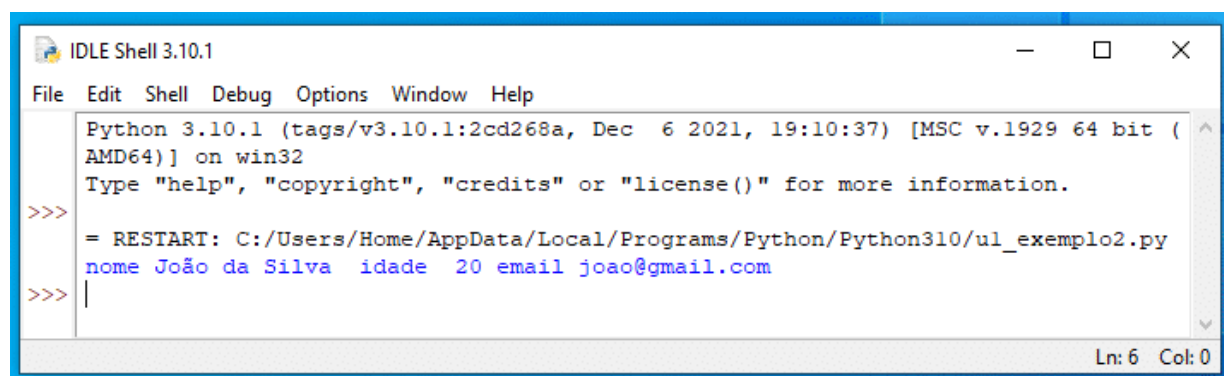
Ln: 3 Col: 55

**Figura 12**

Fonte: Acervo do Conteudista

---

Saída:



```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo2.py
nome João da Silva idade 20 email joao@gmail.com
>>> |
```

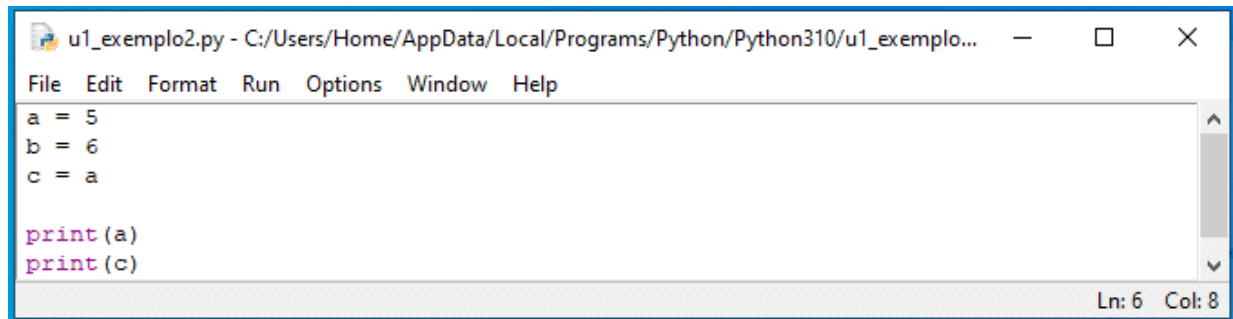
Ln: 6 Col: 0

**Figura 13**

Fonte: Acervo do Conteudista

---

Podemos também atribuir valores às variáveis de acordo com o valor armazenado em outra variável. Por exemplo:



The screenshot shows a window titled 'u1\_exemplo2.py - C:/Users/Home/AppData/Local/Programs/Python/Python310/u1\_exemplo...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

```
a = 5
b = 6
c = a

print(a)
print(c)
```

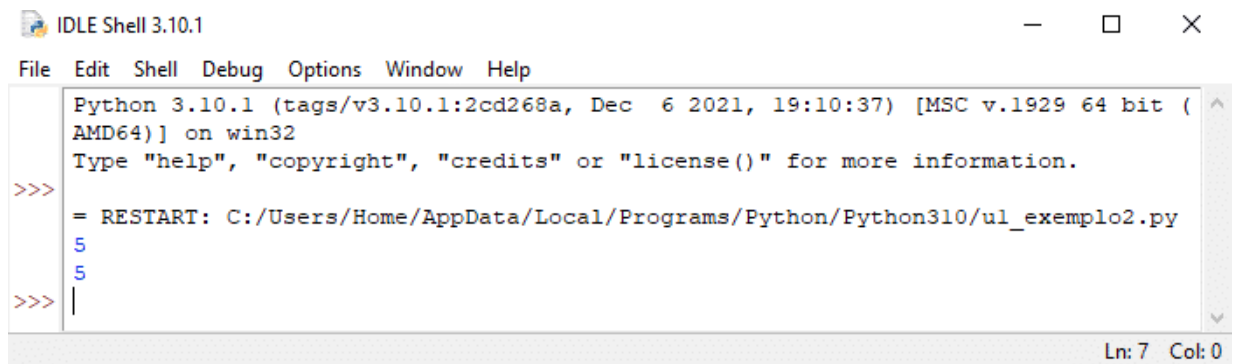
The status bar at the bottom right indicates 'Ln: 6 Col: 8'.

## Figura 14

Fonte: Acervo do Conteudista

- Na linha 3: Atribuímos para a variável c o valor contido na variável a, ou seja, o valor armazenado na variável c será 5, o mesmo valor armazenado na variável a.

Saída:



The screenshot shows the IDLE Shell window titled 'IDLE Shell 3.10.1'. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell displays the following output:

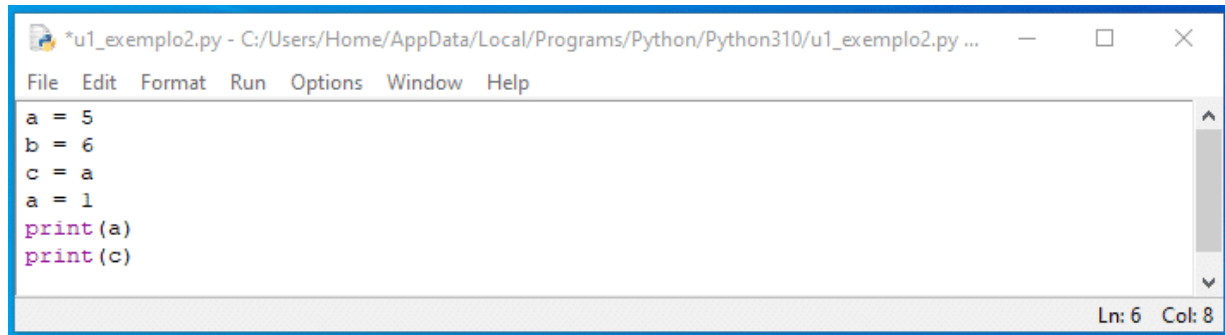
```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo2.py
5
5
>>> |
```

The status bar at the bottom right indicates 'Ln: 7 Col: 0'.

## Figura 15

Fonte: Acervo do Conteudista

É importante ressaltar que a cada vez que é utilizado o sinal de = para atribuir um valor a uma variável, caso tenha nessa variável algum valor atual, o valor será atualizado. Por exemplo:



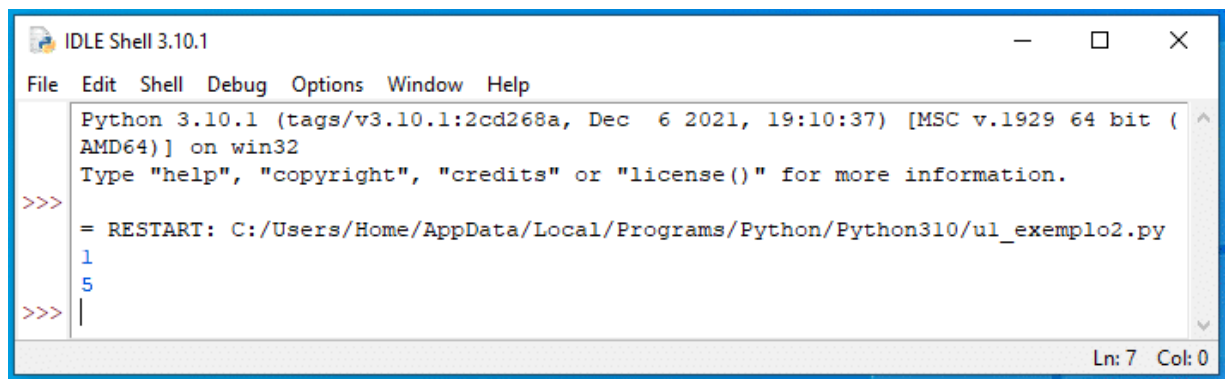
```
*u1_exemplo2.py - C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo2.py ...
File Edit Format Run Options Window Help
a = 5
b = 6
c = a
a = 1
print(a)
print(c)
Ln: 6 Col: 8
```

## Figura 16

Fonte: Acervo do Conteudista

O valor que será exibido para a variável “a” será 1, pois embora na linha 1 tenha sido atribuído o valor 5, na linha 4 foi atribuído o valor 1, uma vez que o *Python* executa linha a linha suas instruções. Já o valor da variável “c” será 5, pois no momento que foi atribuído o valor para a variável “c”, o valor da variável “a” ainda era 5.

Saída:



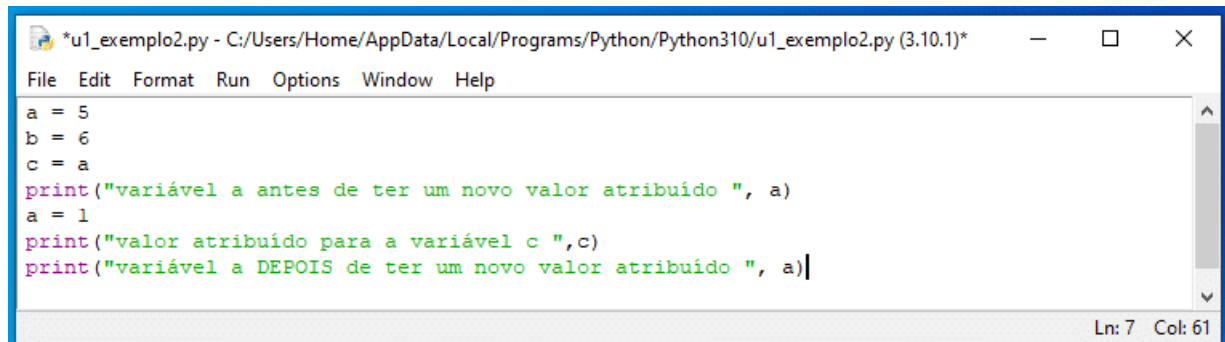
```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo2.py
1
5
>>> |
Ln: 7 Col: 0
```

## Figura 17

Fonte: Acervo do Conteudista

---

Vejamos o mesmo exemplo, porém exibindo os resultados antes e depois da nova atribuição para a variável “a”.



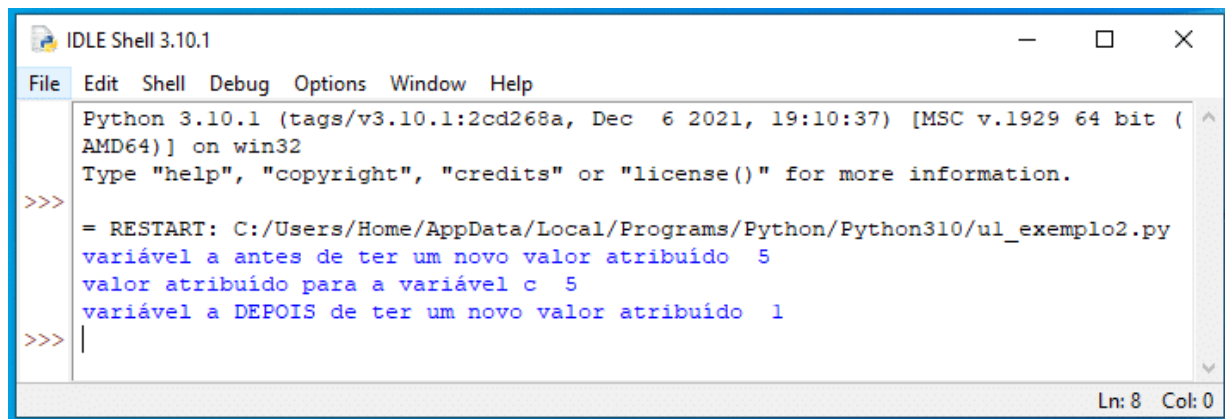
```
*u1_exemplo2.py - C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo2.py (3.10.1)*
File Edit Format Run Options Window Help
a = 5
b = 6
c = a
print("variável a antes de ter um novo valor atribuído ", a)
a = 1
print("valor atribuído para a variável c ", c)
print("variável a DEPOIS de ter um novo valor atribuído ", a)|
Ln: 7 Col: 61
```

## Figura 18

Fonte: Acervo do Conteudista

---

Saída:



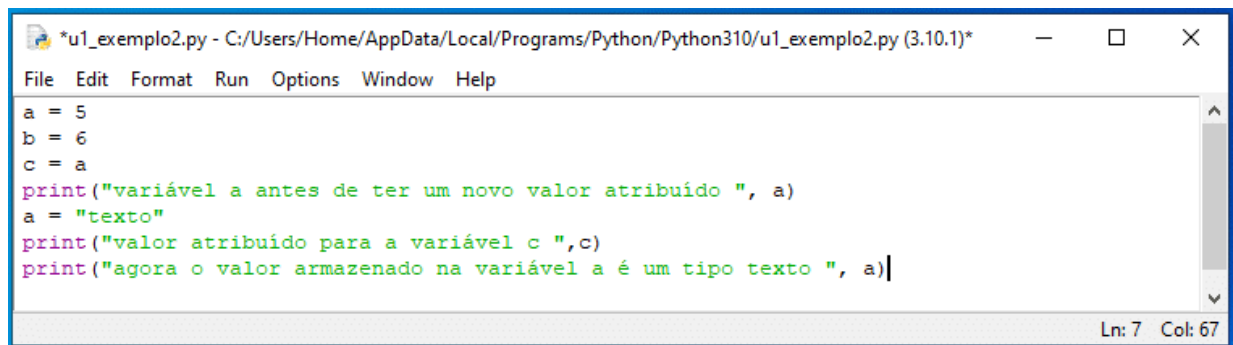
```
IDLE Shell 3.10.1
File Edit Shell Debug Options Window Help
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo2.py
variável a antes de ter um novo valor atribuído 5
valor atribuído para a variável c 5
variável a DEPOIS de ter um novo valor atribuído 1
>>> |
Ln: 8 Col: 0
```

## Figura 19

Fonte: Acervo do Conteudista

---

Por conta da característica de tipagem dinâmica, uma mesma variável pode ter diversos tipos diferentes de dados atribuídos no mesmo programa. Vejamos um exemplo:



```
*u1_exemplo2.py - C:/Users/Home/AppData/Local/Programs/Python/Python310/u1_exemplo2.py (3.10.1)*
File Edit Format Run Options Window Help
a = 5
b = 6
c = a
print("variável a antes de ter um novo valor atribuído ", a)
a = "texto"
print("valor atribuído para a variável c ", c)
print("agora o valor armazenado na variável a é um tipo texto ", a)|
Ln: 7 Col: 67
```

## Figura 20

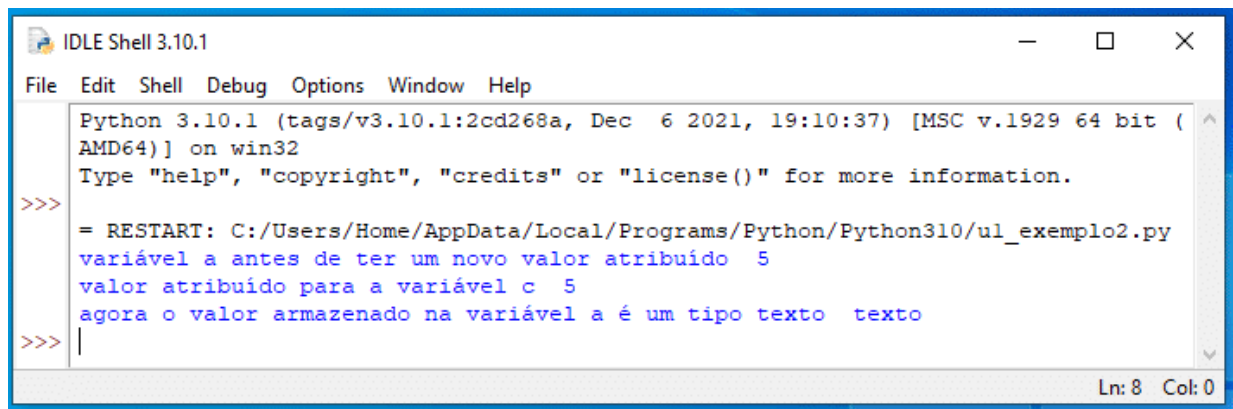
Fonte: Acervo do Conteudista

---

- **Na linha 1:** Atribuímos o valor **5** do tipo número inteiro para a variável “a”;
- **Na linha 5:** Atribuímos o valor **texto** do tipo cadeia de caracteres (*string*) para a variável “a”.

O último valor de saída (impresso na tela) para a variável “a” será o valor “texto”.

Saída:



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/ul_exemplo2.py
variável a antes de ter um novo valor atribuído 5
valor atribuído para a variável c 5
agora o valor armazenado na variável a é um tipo texto texto
>>> |
```

**Figura 21**

Fonte: Acervo do Conteudista

## Exemplos Práticos: Comandos de Entrada de Dados

### Exemplo 1

Em muitas soluções, devemos solicitar dados ao usuário do programa. Para isso, a partir de agora iremos utilizar a função “*input*” em nossos programas sempre que desejarmos que o usuário insira um dado.

A sintaxe dessa função é:



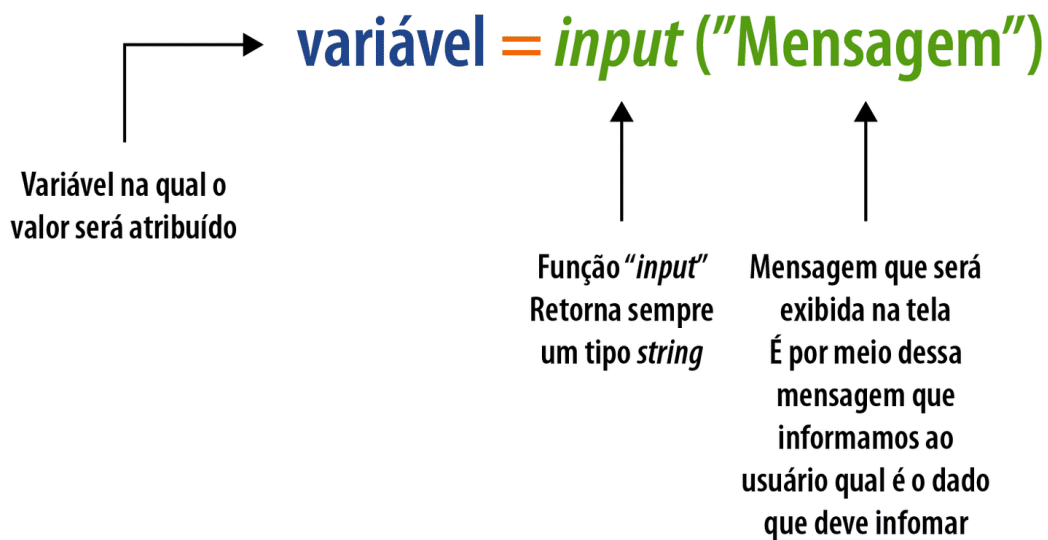


Figura 22

Por exemplo, vamos criar um programa que solicita o nome do usuário e exibe na tela uma saudação com seu nome.

Digite o seguinte código no editor de *scripts* do IDLE:

```
nome = input("digite seu nome")
print("Olá, ", nome, " tudo bem?")
```

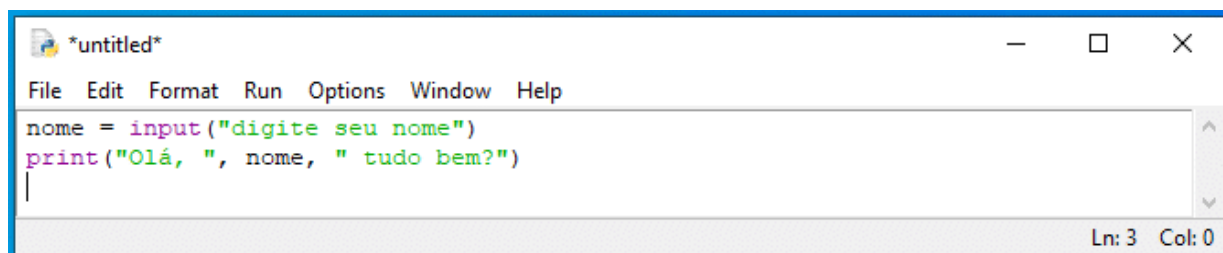
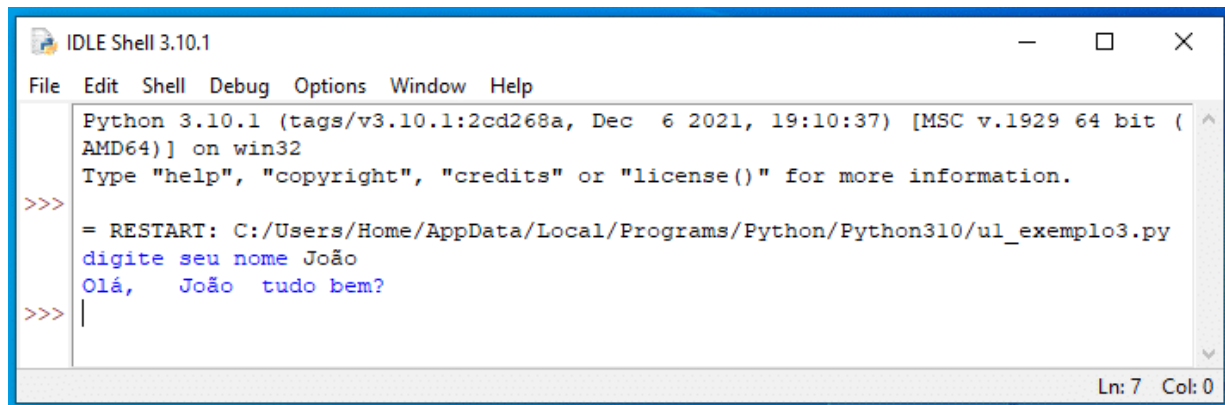


Figura 23

Saída:



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/ul_exemplo3.py
digite seu nome João
Olá, João tudo bem?
>>> |
```

**Figura 24**

Fonte: Acervo do Conteudista

---

## Explicando o Código

- **Na linha 1:** Atribuímos o valor que será informado pelo usuário do programa à variável `nome`. A função `input` sempre retorna um tipo cadeia de caracteres (*string*);
- **Na linha 2:** Por meio da função “`print`”, exibimos (impresso na tela) os valores armazenados na variável `nome`. Concatenamos o valor da variável com os valores fixos digitados (“olá e tudo bem?”).

## Exemplo 2

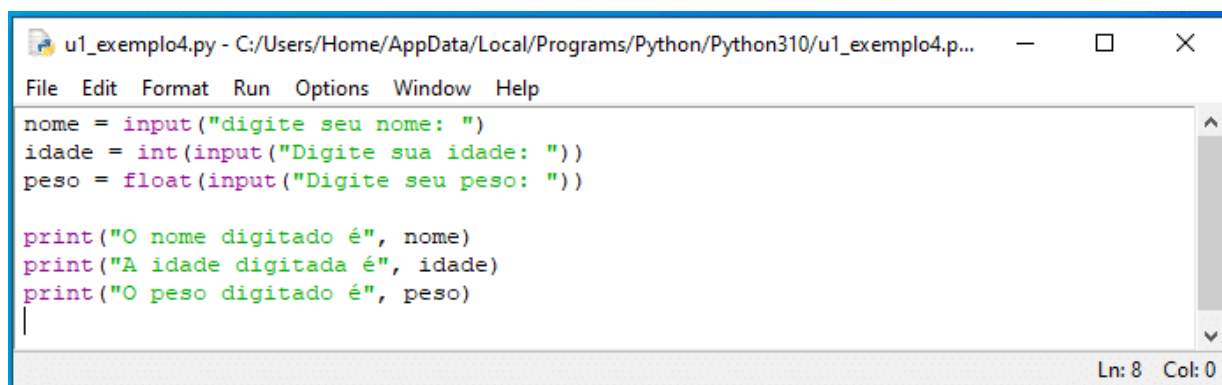
Em muitos cenários, como quando precisamos manipular números para efetuar operações aritméticas, precisamos de alguma forma armazenar esses dados como um tipo número. Sabemos que a função “*input*” sempre retorna um tipo *string*. A estratégia será converter o valor vindo da função “*input*”.

Em nosso segundo exemplo, iremos desenvolver um programa que irá solicitar ao usuário seu nome, idade e altura. Para cada variável, iremos armazenar como tipos diferentes: *string*, inteiro e real.

Digite o seguinte código no editor de *scripts* do IDLE:

```
nome = input("digite seu nome: ")
idade = int(input("Digite sua idade: "))
peso = float(input("Digite seu peso: "))

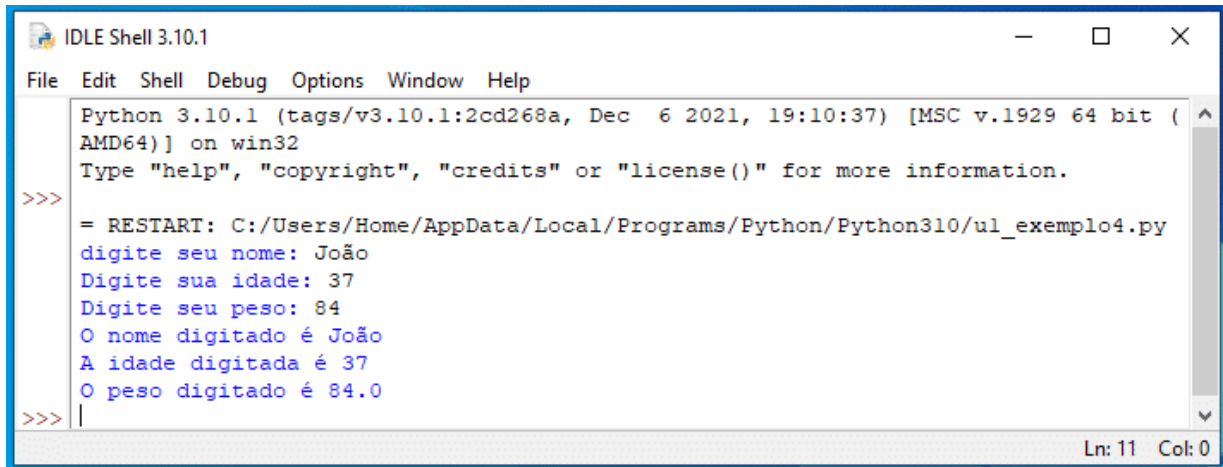
print("O nome digitado é", nome)
print("A idade digitada é", idade)
print("O peso digitado é", peso)
```



**Figura 25**

Fonte: Acervo do Conteudista

Saída:



```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Home/AppData/Local/Programs/Python/Python310/ul_exemplo4.py
digite seu nome: João
Digite sua idade: 37
Digite seu peso: 84
O nome digitado é João
A idade digitada é 37
O peso digitado é 84.0
>>> |
```

**Figura 26**

Fonte: Acervo do Conteudista

## Explicando o Código

- **Na linha 1:** Atribuímos o valor que será informado pelo usuário do programa à variável **nome**. A função *input* sempre retorna um tipo cadeia de caracteres (*string*);
- **Na linha 2:** Atribuímos o valor que será informado pelo usuário do programa à variável **idade**. Porém, diferentemente da primeira linha, convertemos o resultado da função *input* para o tipo inteiro por meio da cláusula *int()*;
- **Na linha 3:** Atribuímos o valor que será informado pelo usuário do programa à variável **idade**. Convertemos o resultado da função *input* para o tipo real (*float*) por meio da cláusula *float()*.

A sintaxe para a conversão dos dados da função *input* para os tipos de número inteiro e real, é:

Converte a entrada de dados para  
um tipo *int* (Números Inteiros)




**variável = int(input("Mensagem"))**

The diagram shows a text label at the top: "Converte a entrada de dados para um tipo *int* (Números Inteiros)". A black arrow points from this label down to the *int* function in the code snippet below. The code snippet is "variável = int(input("Mensagem"))", where "variável" is in blue, "=" is in orange, "int" is in blue, "input" is in green, and "Mensagem" is in green and enclosed in quotes.

Figura 27

---

Converte a entrada de dados para  
um tipo *float* (Números Reais)



**variável = float(input("Mensagem"))**

The diagram shows a text label at the top: "Converte a entrada de dados para um tipo *float* (Números Reais)". A black arrow points from this label down to the *float* function in the code snippet below. The code snippet is "variável = float(input("Mensagem"))", where "variável" is in blue, "=" is in orange, "float" is in red, "input" is in green, and "Mensagem" is in green and enclosed in quotes.

Figura 28

---

É importante ressaltar que se o usuário digitar um número real (um número com casas decimais, por exemplo) para um *input* convertendo em *int*, o número digitado com casas decimais será armazenado como um inteiro.

Destaca-se também que a tentativa de se converter uma cadeia de caracteres em algum tipo de número acarretará erro de execução do programa.

---

## Em Síntese

Nesta Unidade, além de aprendermos a instalar o ambiente de desenvolvimento *Python*, estudamos os conceitos de identificadores, uso e atribuição de valores de variáveis e comandos de entrada (*input*) e saída (*print*). É importante que assista à videoaula da Unidade e que leia os livros e materiais complementares indicados.



# Material Complementar

---

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

---

## Vídeos

**Como Instalar e Testar o *Python 3.9* no *Windows 10***



## *Python – Variáveis, Tipos de Dados e o Comando Type*





Leitura

## Módulos e Pacotes em *Python*

Clique no botão para conferir o conteúdo.

ACESSE

## O Tutorial de *Python*

Clique no botão para conferir o conteúdo.

ACESSE



## Referências

---

BANIN, S. L. **Python 3: conceitos e aplicações** – Uma abordagem didática. São Paulo: Érica, 2018. (*e-book*)

PERKOVIC, L. **Introdução à Computação Usando Python: um foco no desenvolvimento de aplicações**. Rio de Janeiro: LTC, 2016. (*e-book*)

WAZLAWICK, R. **Introdução a Algoritmos e Programação com Python: uma abordagem dirigida por testes**. Rio de Janeiro: Elsevier, 2017. (*e-book*)