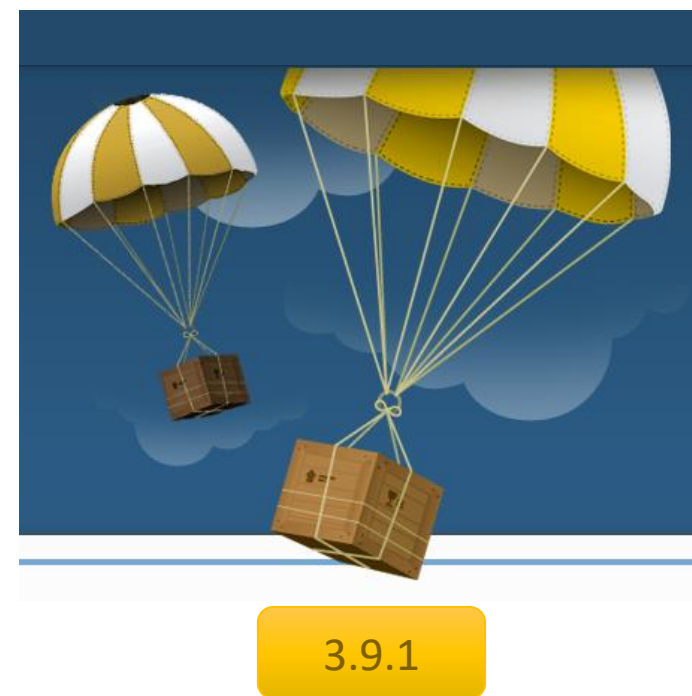




INTRODUÇÃO A PYTHON

≡ Python ?!

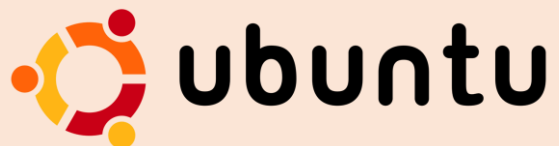
- **Linguagem eficiente e produtiva:**
 - Menor tempo de desenvolvimento;
 - Fácil interpretação;
 - Curva de aprendizagem reduzida.
- **Usada para administrar sistemas e desenvolver grandes projetos;**
- **Software livre:**
 - Mantida pela Python Foundation e inúmeros colaboradores.
- **Multiplataforma:**
 - Linux, Windows, Mac OS, ...
- **Produtividade:**
 - Frameworks!





Download

www.python.org/downloads



Já vem instalado !



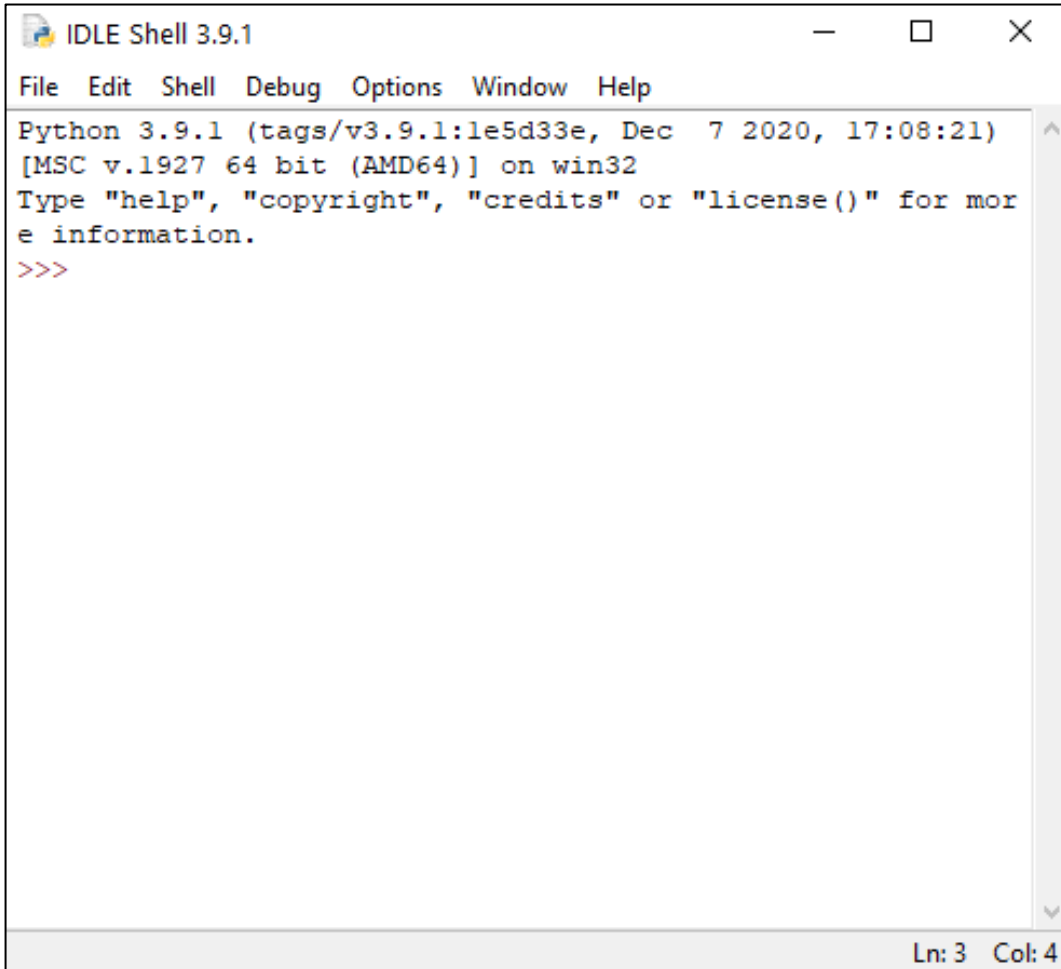
Já vem instalado !



Onde vou programar?

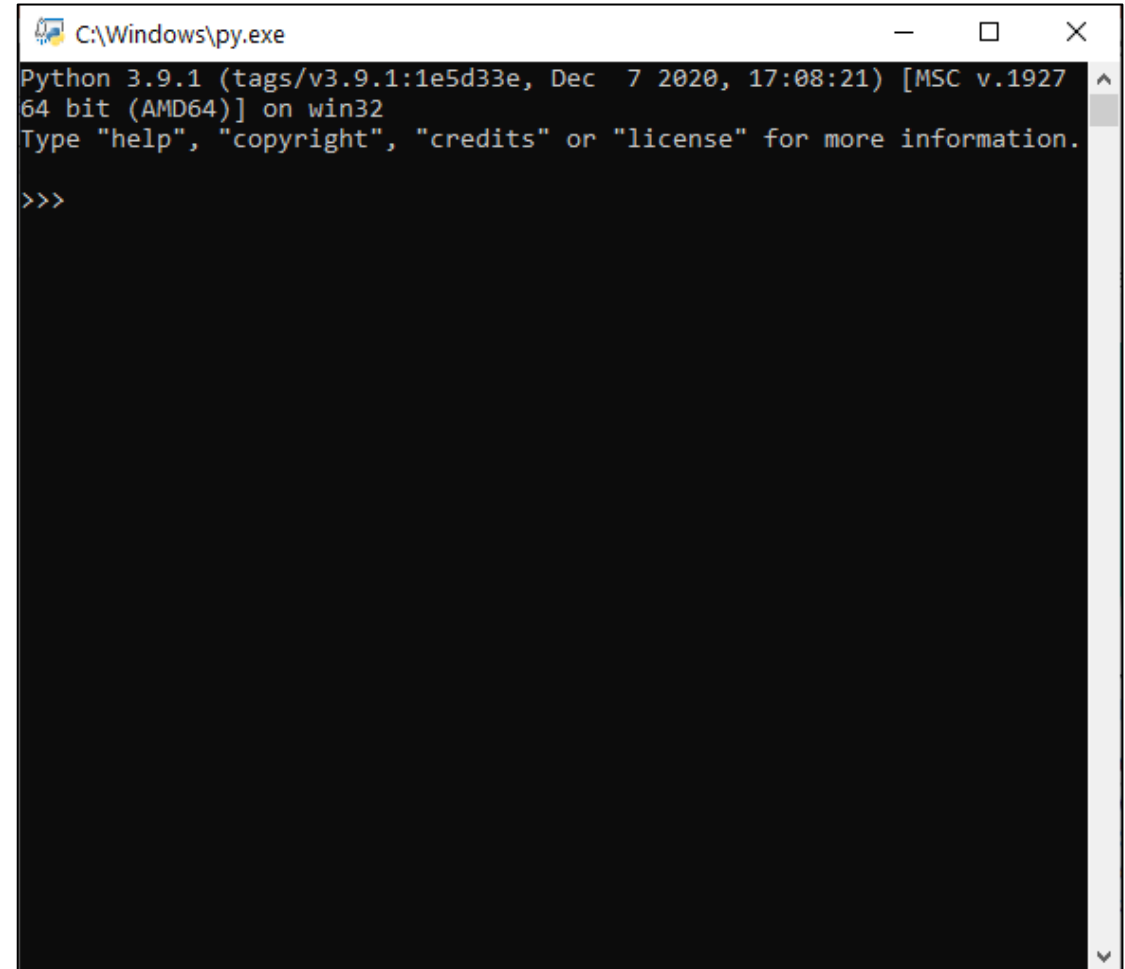


Windows



```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21)
[MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
```

IDLE (Python GUI)



```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Linha de Comandos



Olá, este é o Colaboratory

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto Copiar para o Drive

Conectar Editar

O que é o Colaboratory?

O Colaboratory ou "Colab" permite escrever código Python no seu navegador, com:

- Nenhuma configuração necessária
- Acesso gratuito a GPUs
- Compartilhamento fácil

Você pode ser um **estudante**, um **cientista de dados** ou um **pesquisador de IA**, o Colab pode facilitar seu trabalho. Assista ao vídeo [Introdução ao Colab](#) para saber mais ou simplesmente comece a usá-lo abaixo!

Primeiros passos

O documento que você está lendo não é uma página da Web estática, mas sim um ambiente interativo chamado **notebook Colab** que permite escrever e executar código.

Por exemplo, aqui está uma **célula de código** com um breve script Python que calcula um valor, armazena-o em uma variável e imprime o resultado:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day
```

86400

Para executar o código na célula acima, clique nela e depois pressione o botão Play à esquerda do código ou use o atalho do teclado "Command/Ctrl+Enter". Para editar o código, basta clicar na célula e começar a editar.

As variáveis definidas em uma célula podem ser usadas mais tarde em outras células:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
    seconds_in_a_week
```

604800

≡ Elementos básicos da linguagem Python

- Variável
- Tipos de dados
- Operadores aritméticos
- Atribuição
- Entrada
- Saída

Definição:

- Espaço reservado na memória para armazenamento de dados.

Declaração:

- Para declarar uma variável é necessário definir o identificador e (atribuir) um valor para inicializar.

Tipagem dinâmica:

- Tipos básicos: **int**, **float**, **str** e **bool**.

Declaração!

```
vi = 1  
vf = 1.1  
vs = "ifpb"  
vb = True
```




Algumas regras para definição do “identificador”:

- Não pode começar com número;
- Não pode conter caracteres especiais (exceto _);
- Não pode ser palavra reservada da linguagem;
- Usar letras minúsculas;
- Usar nomes significativos.



Palavras Reservadas

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

≡ Operadores Aritméticos

Operador	Descrição	Exemplo	Resultado
+	Soma	$2 + 2$	4
-	Subtração	$2 - 2$	0
*	Multiplicação	$2 * 2$	4
/	Divisão Real	$2 / 2$	1.0
//	Divisão Inteira	$5 // 2$	2
%	Resto da Divisão	$2 \% 2$	0
**	Potência	$2 ** 2$	4

≡ Atribuição

- Atribui um valor a uma variável
- Sintaxe:

<variável> = <valor>

Exemplos

```
>>> frase = 'Programando em Python'
>>> frase
'Programando em Python'
>>>
>>> n = 5
>>> n
5
>>>
>>> x = n + 2
>>> x
7
```

≡ Entrada: `input()`

- Sempre retorna um valor do tipo “str” (string);

Exemplo: leitura de uma frase

```
>>> frase = input ('Digite uma frase: ')
Digite uma frase: Programação em Python
>>> frase
'Programação em Python'
```

- Mas como Identificar e converter tipos em Python quando for preciso manipular valores numéricos?

Exemplo: somando dois números

```
>>> numero1 = input ('Digite um numero: ')
Digite um numero: 2
>>> numero2 = input ('Digite um numero: ')
Digite um numero: 3
>>> soma = numero1 + numero2
>>> soma
'23'
```

≡ Identificação e Conversão de Tipos

- Identificando um tipo:
 - `type (<valor ou variável>)`

Exemplo

```
>>> type(soma)
<class 'str'>
```

Em relação ao exemplo anterior, soma é do tipo string

- Funções de Conversão
 - `int()`
 - `float()`
 - `str()`

Exemplo 1

```
>>> soma = int(numero1) + int(numero2)
>>> soma
5
```

Exemplo 2

```
>>> numero1 = int(input('Digite um número: '))
Digite um número: 2
>>> type(numero1)
<class 'int'>
```

≡ Saída: *print()*

- `print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)`

sep	O valor padrão de sep é um espaço em branco, quando dois ou mais argumentos são passados para a função print sep coloca entre eles um espaço em branco ou um outro valor que podemos passar para sep.
end	O valor padrão de end é uma nova linha “\n” e por isso que sempre a função print adiciona uma nova linha depois de imprimir tudo que lhe foi passado, mas, podemos definir outro valor com uma tabulação “\t” por exemplo.

Exemplo

```
nome=input('Informe um nome:')  
print('Nome =', nome)
```

≡ Apresentando dados em Python com *f-strings*

- Pode-se usar a partir da versão 3.6 do Python.
- O nome *f-string* é devido ao prefixo *f* que deve existir junto à string.

```
>>> aluno = 'Pedro'
>>> curso = 'Sistemas para Internet'
>>> print(f'{aluno} está cursando {curso}')
Pedro está cursando Sistemas para Internet
```

```
>>> n = 3
>>> print(f'O dobro de {n} é {n*2}')
O dobro de 3 é 6
```


≡ Apresentando dados em Python com *f-strings*

- Exemplos de formatação de strings:

```
>>> texto = 'Aula de APE'
>>>
>>> print(f'{texto:20}')
Aula de APE
>>>
>>> print(f'{texto:>20}')
      Aula de APE
>>>
>>> print(f'{texto:<20}')
Aula de APE
>>>
>>> print(f'{texto:^20}')
      Aula de APE
>>>
>>> print(f'{texto:*^20}')
****Aula de APE****
```

≡ Apresentando dados em Python com *f-strings*

- Exemplos de formatação de números:

```
>>> inteiro = 54
>>> decimal = 54.79
>>>
>>> print(f'{inteiro} {decimal:.2f}')
54 54.79
>>>
>>> print(f'{inteiro:8} {decimal:8.2f}')
      54      54.79
>>>
>>> print(f'{inteiro:+} {decimal:+.2f}')
+54 +54.79
>>>
>>> print(f'{inteiro:08} {decimal:08.2f}')
00000054 00054.79
```