

Python é uma linguagem de programação gerenciada pela Python Software Foundation. Essa linguagem é de alto nível disponível em código aberto (open source).

O que você pode fazer com Python?

- Acelerar o aprendizado do conteúdo de cálculo computacional
- Construir sistemas baseados em Web com frameworks
- Analizar dados
- Criar sistemas que se acoplem em projetos de inteligência artificial
- Aplicar técnicas de bibliotecas em Machine Learning
- Facilitar a criação de aplicativos
- Construir sistemas para desktop

O Python possui bibliotecas padronizadas dotadas de muitos recursos, que atraem boa parte dos programadores.

A arquitetura do python possui frameworks e módulos que são implementados e atualizados pela comunidade que contribui com o projeto.

Em que o Python nos ajudará?

- *Auxiliará o profissional a resolver problemas reais utilizando os recursos computacionais.*
- *Facilitará a implementação da modelagem dos problemas para testar as soluções computacionais.*
- *Ajudará nas soluções computacionais por meio de algoritmos e sua correta execução.*
- *Suportará soluções via software de modo a garantir consistências nos algoritmos de cálculos em fluxos dinâmicos de grandes quantidades de dados.*
- *Suportará a disciplina de cálculo computacional.*

O Python é uma linguagem de programação de alto nível (figura1 ). LP são linguagens formais, com um teor preciso e instruções, estas últimas, são executadas em um sistema computacional.

Um código fonte é um conjunto de palavras associadas as regras específicas da linguagem que está sendo utilizada para a programação.

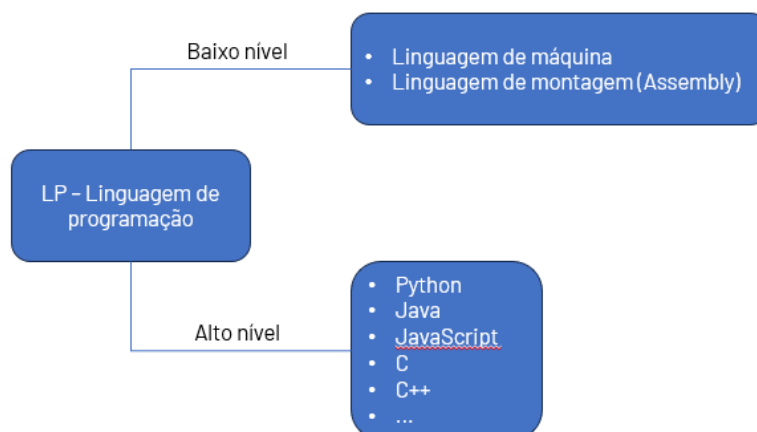


Figura 1: Python é uma linguagem de alto nível (autoral)

Esse conjunto passa por um processo que traduz em uma linguagem reconhecida pelo computador. Isso pode ocorrer em três abordagens, a abordagem está relacionada a propriedade da sua implementação (figura 2).

- Linguagem Interpretada
- Linguagem Compilada
- Linguagem com Byte-Code

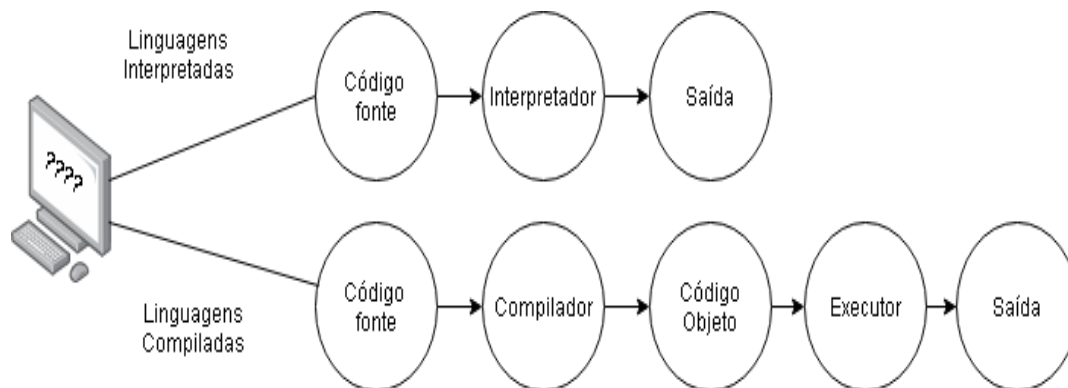


Figura 2: Propriedades das implementações das abordagens das linguagens de programação. (autoral)

A abordagem da **Linguagem Interpretada** tem as seguintes características:

1. é transformada em linguagem de máquina em tempo de execução do código.
2. o Interpretador de código é carregado na memória da máquina para execução.
3. a memória é gerenciada.

## Explicando melhor

Na abordagem em linguagem interpretada, um código fonte, como um arquivo é escrito com determinados comandos relativos a linguagem de alto nível. Esse código fonte é dado como entrada em outro programa, que é o interpretador. Esse interpretador segue linha a linha do arquivo fonte. Cada linha de código é interpretada e algo é executado internamente no computador. E pode gerar algum resultado na saída do terminal do computador, encaminhado para a impressora, algum tipo de bytes para a internet, dentre outras interações de saída.

A abordagem da linguagem compilada tem as seguintes características:

1. Traduzir as instruções do código fonte em uma linguagem de máquina na etapa de compilação.
2. O código executável gerado não interpreta a linguagem, apenas contém a linguagem de máquina.
3. A execução é mais rápida do que linguagens interpretadas. O gerenciamento do uso da memória fica por conta do programador.

## Explicando melhor

Na linguagem compilada, o código fonte, o programa, é para o compilador, esse compilador transforma o programa em código de máquina. Esse código de máquina são os zeros(0) e uns(1). Esse código objeto é gravado em disco como um arquivo executável. Posteriormente esse arquivo executável, depois que a compilação terminou, pode ser executado, por um programa executor, tipicamente é próprio sistema operacional. O sistema operacional vai executar esse arquivo “executável” que vai realizar alguma interação com o usuário ou gerar uma saída.

As linguagens com Byte-Code usam as duas abordagens, linguagens interpretadas e compiladas. Linguagens modernas combinam as duas abordagens como o Java e Python. Há características das duas abordagens anteriores.

No processo de compilação, é gerado um arquivo intermediário, denominado de byte-code, não gera uma linguagem de máquina. Antes de iniciar a execução do programa, um compilador traduz o código-fonte para byte-code, são códigos em bytes.

No processo de execução, o byte-code é interpretado, gerando assim a linguagem de máquina. Ao iniciar a execução do programa, o interpretador lê os bytecodes um a um executando os comandos correspondentes.

A interpretação do byte-code é realizada a primeira vez em que for executado, permitindo execuções posteriores mais rápidas.

Nesta abordagem a memória é gerenciada.

## Explicando melhor

Antes de iniciar do programa, o compilador traduz o código fonte desse programa, para um código de bytes, que chamamos de byte-codes. Os byte-codes são uma forma reduzida do mesmo programa só que de maneira mais fácil para que o computador possa interpretá-la. Após isso, o byte-code, arquivo pré-compilado, é submetido a um interpretador que lê eficientemente os byte-codes e o executando um a um os comandos correspondentes.

**Instalando o Python em sua máquina. Escolha seu SO. Neste e book estamos com Windows10.**

<https://www.python.org/downloads/>

<https://www.python.org/>



Clique no download abaixo ou na sua área específica de downloads.

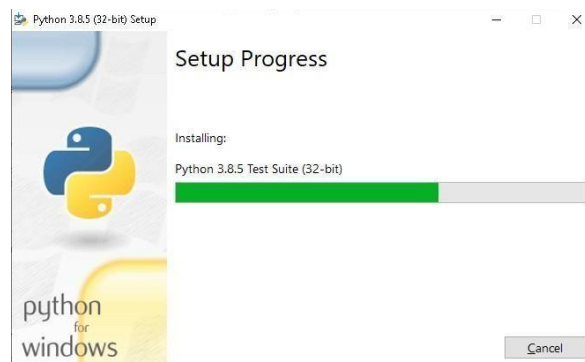
Instalação  
nova



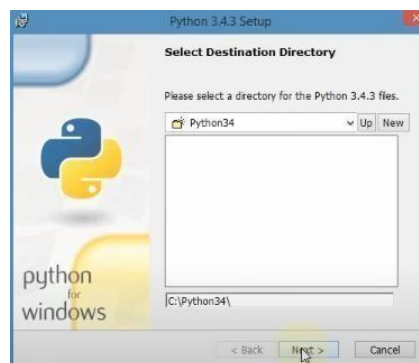
Upgrade quando há alguma  
versão instalada



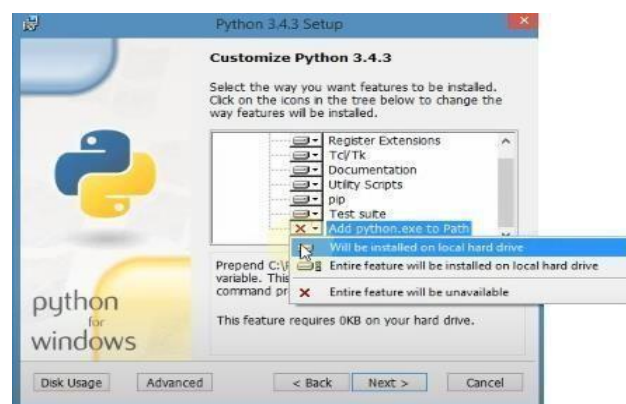
Como já tenho o Python instalado farei um upgrade.



Em alguns casos precisa selecionar o caminho (path), clic em next



Selecione como indicado a seguir

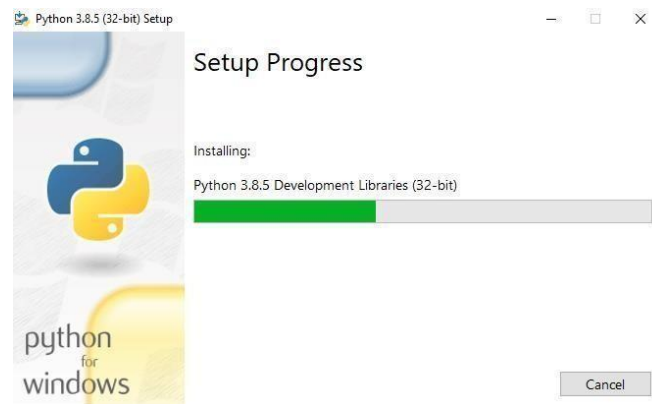


Após clique em next

Agora pode aparecer um aviso de segurança no Windows

Concorde se o fornecedor verificado for da Python Software Foundation, clique em sim.

Agora começa a Instalação:



Aguarde a barra de progresso até aparecer finish

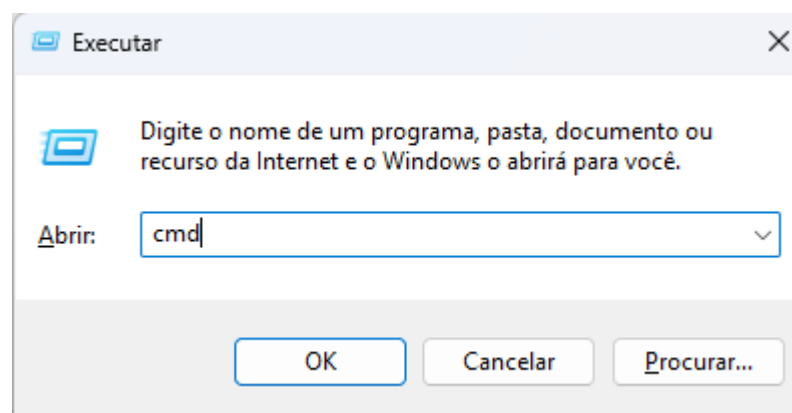
Clique em close:



Clique em disable, por causa do limit do tamanho path. Agora vamos executar o Python a partir da linha de comando.

Vamos abrir o interpretador de comandos Python

Aperte: **Win + R**



5

```
Prompt de Comando
Microsoft Windows [versão 10.0.22621.1992]
(c) Microsoft Corporation. Todos os direitos reservados.
C:\Users\Eduardo Verri>
```

**cmd** é o modo de linha de comandos (CLI) do Windows

Se tudo deu certo agora vamos digitar **python** ou **py**

```
Prompt de Comando - pythor
C:\Users\Eduardo Verri>python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

Agora vc já tem o interpretador de comandos!

Então vamos agora utilizar o interpretador. Faça cada comando, e em seguida tecle **Enter**:

```
>>> a=1
>>> b=2
>>> c=a+b
>>> c
3
>>> a=4
>>> b=2
>>> c=a-b
>>> c
2
>>> a=3
>>> b=2
>>> c=a*b
>>> c
6
>>> a=8
>>> b=4
>>> c=a/b
>>> c
2.0
```

Cuidado com a **indentação**<sup>1</sup> em Python

Vamos fazer um exemplo

```
>>> a = 1
File "<stdin>", line 1 a=1
^
```

<sup>1</sup> Indentação é empregada para ressaltar a estrutura do algoritmo

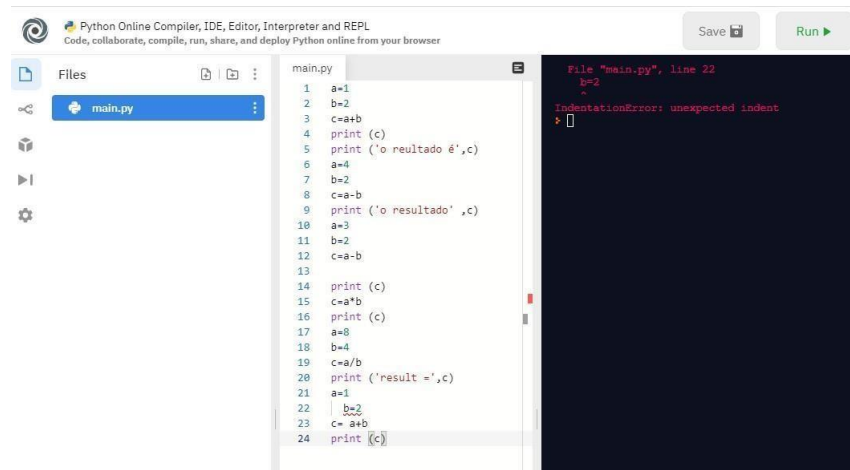
**IndentationError: unexpected indent**

Agora vamos fazer as mesmas operações em ambiente completo, mas on line



Python Online Compiler, IDE, Editor, Interpreter and REPL  
Code, collaborate, compile, run, share, and deploy Python online from your browser

"read-evaluate-print loop" – REPL em <https://repl.it/languages/python3>



Vamos agora com operações mais elaboradas

A princípio vamos "limpar o console"

```
>>>print('\n'*100)
>>># Atribuir valores inteiros a duas variáveis
>>>a = 6
>>>b = 3
>>># Vamos fazer operações mais elaboradas usando as 4 operações simples
>>>soma = a+b
>>>subtracao =a-b
>>>multiplicacao = a*b
>>>divisao =a/b
>>>print("Vamos mostrar todos os resultados") print("a=", a)
>>>print("b=", b)
>>>print("a +b=", soma)
>>>print("a -b=", subtracao)
>>>print("a *b=", multiplicacao)
>>>print("a / b=", divisao)
```

O que podemos observar em relação aos números inteiros e a saída resultante da divisão?

Agora vamos trabalhar com números decimais ou tipo ponto flutuante

```
>>>print('\n'*50)
>>>a1=6.5
>>>b1=2.5
>>># Vamos fazer operações mais elaboradas usando as 4 operações simples
>>>soma = a1+b1
>>>subtracao =a1-b1
>>>multiplicacao = a1*b1
```

```
>>divisao =a1/b1
>>print('Vamos mostrar todos os resultados')
>>print('a1=', a1)
>>print('b1=',b1)
>>print('a1 +b1=', soma)
>>print('a1-b1=', subtracao)
>>print('a1*b1=',multiplicacao)
>>print('a1 / b1=',divisao)
```

Se houver algum erro apontado pelo interpretador nas operações acima **corrija**.

Vamos utilizar técnicas simples para duas operações complexas, a **radiciação e a potenciação**.  
Nós poderíamos importar biblioteca específica para isso, mas não faremos neste momento.

### Radiciação

Lembra disso  $\sqrt{25}$  ?

Este símbolo é a raiz quadrada não negativa ou raiz principal.

Então o resultado dessa raiz será um número x multiplicado por ele mesmo. Se:

$$\sqrt{25} = \sqrt{5 * 5} = \sqrt{5 ** 2} = (5^2)^{\frac{1}{2}} = 5^{\frac{2}{2}} = 5$$

$$\sqrt[3]{27} = \sqrt[3]{3 * 3 * 3} = \sqrt[3]{3 ** 3} = (3^3)^{\frac{1}{3}} = 3^{\frac{3}{3}} = 3$$

Parece confuso, vamos recordar a potenciação, isto vai ajudar a compreender a radiciação.

A forma utilizada é \*\*

- $a**b = a^b$ , a elevado a b, onde a é a base e b é o expoente
- $(a**x1)*(a**x2) = a^{x1+x2}$ , conserva a base e soma os expoentes
- $(a**2) = a^2$ , a elevado ao quadrado
- $(a**3) = a^3$ , a elevado ao cubo
- $(a**4) = a^4$ , a elevado a quarta
- $(a**n) = a^n$ , a elevado a enésima potência

### Exercício1) Faça um código em Python

Se a=2, atribua:

```
potencia_ao_quadrado = a**2
```

```
potencia_ao_cubo = a**3
```

```
potencia_a_quarta = a**4
```

Imprima o resultado das variáveis

Agora vamos usar a biblioteca math, importando a função pow (potência)

Faça em Python

```
>>from math import pow
>>print('\n'*100)
>># Atribua valores para c e d
>>c = 4
>>d = 5
>>c_elevado_ao_quadrado = pow(c,2)
```



```
>>c_elevado_ao_cubo = pow(c,3)
>>c_elevado_a_quarta = pow(c,4)
>>c_elevado_a_d = pow(c,d)
>>print("c elevado ao quadrado =", c_elevado_ao_quadrado)
>>print("c elevado ao cubo =", c_elevado_ao_cubo)
>>print("c elevado a quarta =", c_elevado_a_quarta)
>>print("c elevado a d =", c_elevado_a_d)
```

Agora é possível exercitar a radiciação sem uso de biblioteca

**Exercício2)** Faça um código em Python

Se x=512, atribua:

raiz\_quadrada\_de\_x

raiz\_cúbica\_de\_x

raiz\_quarta\_de\_x =

Imprima o resultado das variáveis

**Arredondamentos e operações inteiras** por meio de biblioteca math.

Você já ouviu falar de piso e teto? Mas no contexto numérico? Então, vai uma dica, piso salarial e teto salarial.

## Explicando melhor

O **piso salarial** é o menor valor de salário que pode ser pago a uma categoria profissional por sua jornada de trabalho. Já o **teto salarial** é o maior salário que pode ser pago para realização de dado serviço. É o oposto de piso salarial.

Dado um salário  $w = 3345.61$ . Qual o seu teto e piso salarial?

Para responder a esta questão que poderá ser muito útil para números decimais também, vamos usar três funções do Python.

- floor – retorna o maior valor inteiro menor que w
- ceil – retorna o menor valor inteiro maior que w
- round – retorna o valor inteiro mais próximo de w

**Exercício 3)** dado o valor de w anteriormente, importe as funções floor, ceil da biblioteca math e determine o piso, o teto e o arredondamento.

**Exercício 4)** Porque não importamos o round? Faça uma pesquisa nos sources “round não é função em python”. Vamos ver como usar o round (número, n\_digits).

```
>>w= 3345.61
>>round(w,1)
```

Explique a saída da função anterior, e se usássemos round(w,0)?

**Exercício 5)** Arredonde os números a seguir, deixando n\_digits = none

```
>>x1=1.456
```

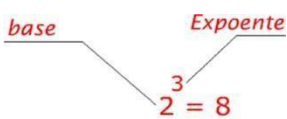
```
>>x2=3.678
>>x3=7.5
```

**Exercício 6)** Voltando ao caso do floor e ceil, por padrão estas funções retornam valores decimais, ou seja, com ponto flutuante. Podemos determinar que ambas funções resultem em uma saída com inteiro mantendo a característica de piso e teto. Faça o teste com o comando a seguir:

```
>>int(floor(1.456))
```

**Exercício 7)** Resolva em Python a expressões a seguir:

Lembre-se



Dadas as Propriedades a seguir como você resolveria a lista de exercícios de potenciação em Python?

Propriedades	Exercícios
<ol style="list-style-type: none"> <li><math>a^b = a \times a \times a \dots a</math> (b vezes)</li> <li><math>a^0 = 1</math></li> <li><math>a^1 = a</math></li> <li><math>a^{-b} = \frac{1}{a^b}</math>, <math>a \neq 0</math></li> <li><math>a^b \times a^c = a^{b+c}</math></li> <li><math>\frac{a^b}{a^c} = a^{b-c}</math>, <math>a \neq 0</math></li> <li><math>(a^b)^c = a^{b \times c}</math></li> <li><math>\left(\frac{a}{b}\right)^c = \frac{a^c}{b^c}</math>, <math>b \neq 0</math></li> </ol>	<p>Calcule o valor das expressões:</p> <ol style="list-style-type: none"> <li><math>2^3</math></li> <li><math>(-2)^3</math></li> <li><math>1^0</math></li> <li><math>(-1)^0</math></li> <li><math>2^0</math></li> <li><math>\left(\frac{2}{5}\right)^3</math></li> <li><math>3^{-2}</math></li> <li><math>\left(\frac{1}{2}\right)^{-3}</math></li> <li><math>((-1)^3)^4</math></li> <li><math>(0,5)^3</math></li> <li><math>(0,25)^4</math></li> <li><math>0^4</math></li> <li><math>1 + (0,41)^2</math></li> <li><math>\frac{1}{4} + 5^2 - 2^{-4}</math></li> <li><math>2^{-3} + (-4)^{-5}</math></li> <li><math>\left(\frac{4}{5} - \frac{1}{2} + 1\right)^{-2} + \frac{1}{1 + 3^2 - (4 - 5)^{-2}}</math></li> </ol>

**Dicas**

1. O Python é case-sensitive, há diferenças entre sum e SUM

2. Entrar e sair do Interpretador Python em cmd:

C:\Users\Eduardo Verri>**python**

Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>exit()

-ou-

C:\Users\Eduardo Verri>**python**

Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>quit()

**Ambos comandos, pertencem a classe quitter**

**Sair do terminal, aperte as teclas a seguir: Ctrl + z dê Enter**

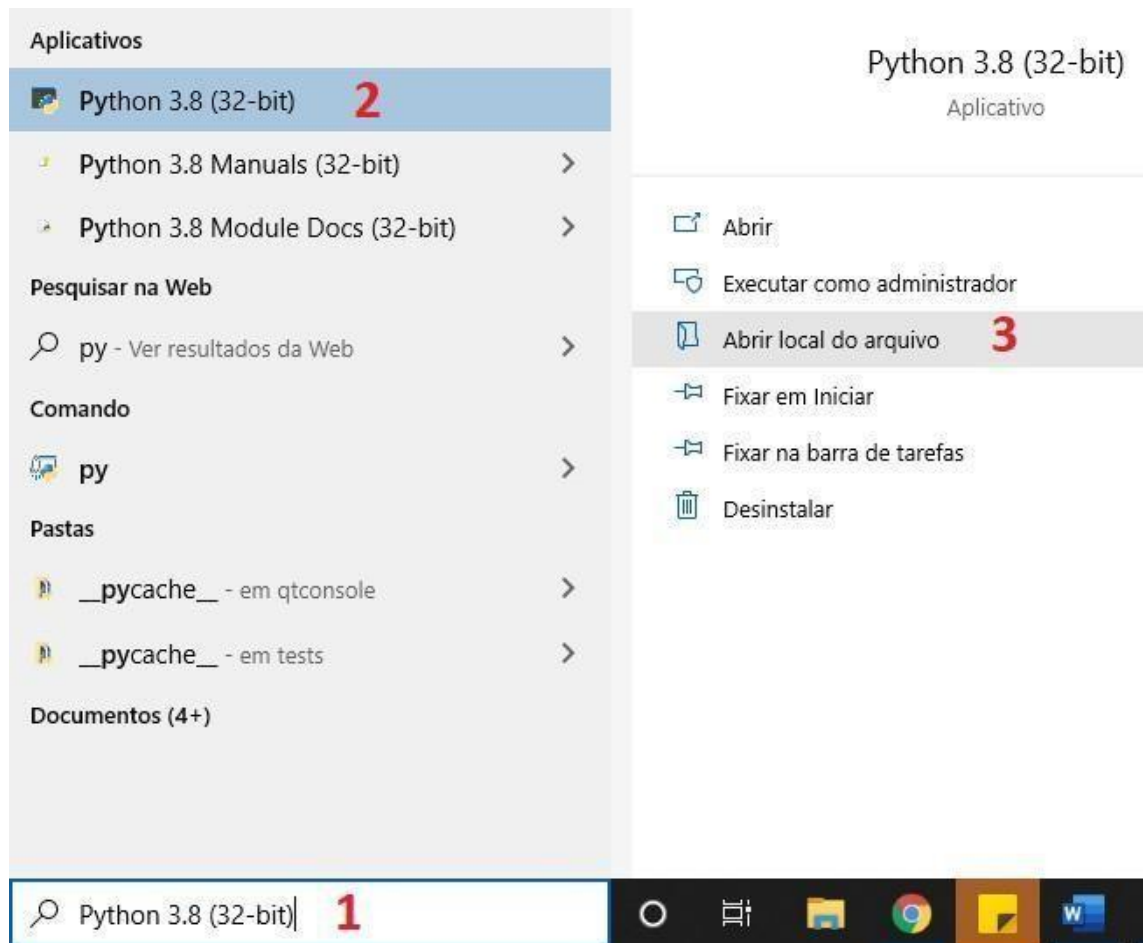
3. O python está instalado e vc digita **py** ou **python** no terminal, mesmo assim ocorre o seguinte erro:

C:\Users\Eduardo Verri>python

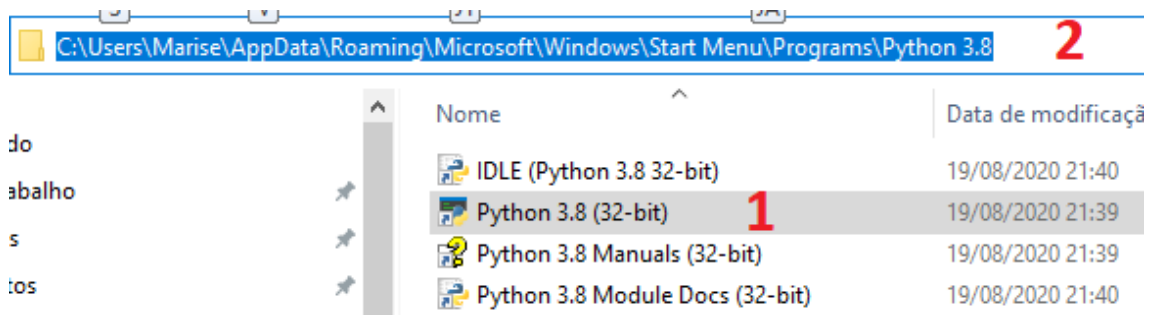
'python' não é reconhecido como um comando interno ou externo, um programa operável ou um arquivo em lotes. Como resolver:

Vamos configurar o python como variável de ambiente

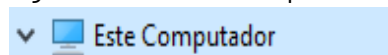
Vá em menu iniciar (1) e digite Python, em aplicativos selecione (2) apenas e clique em 3



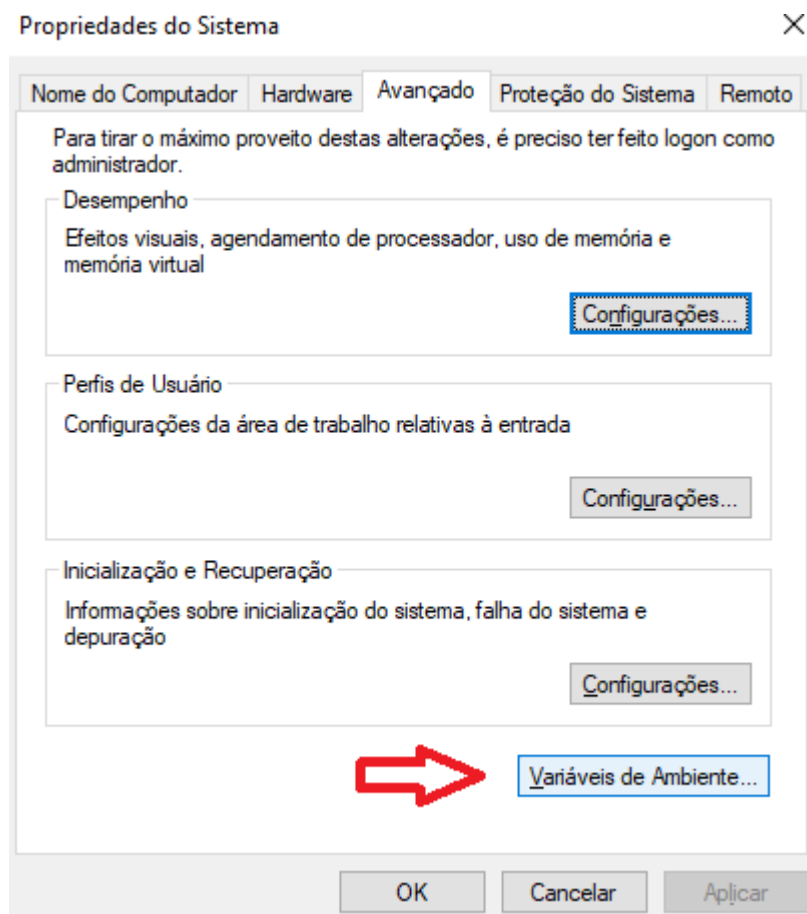
Quando entrar no local em que está armazenado o Python (1), copie o caminho do executável (2). Conforme passos 1 e 2 a seguir:



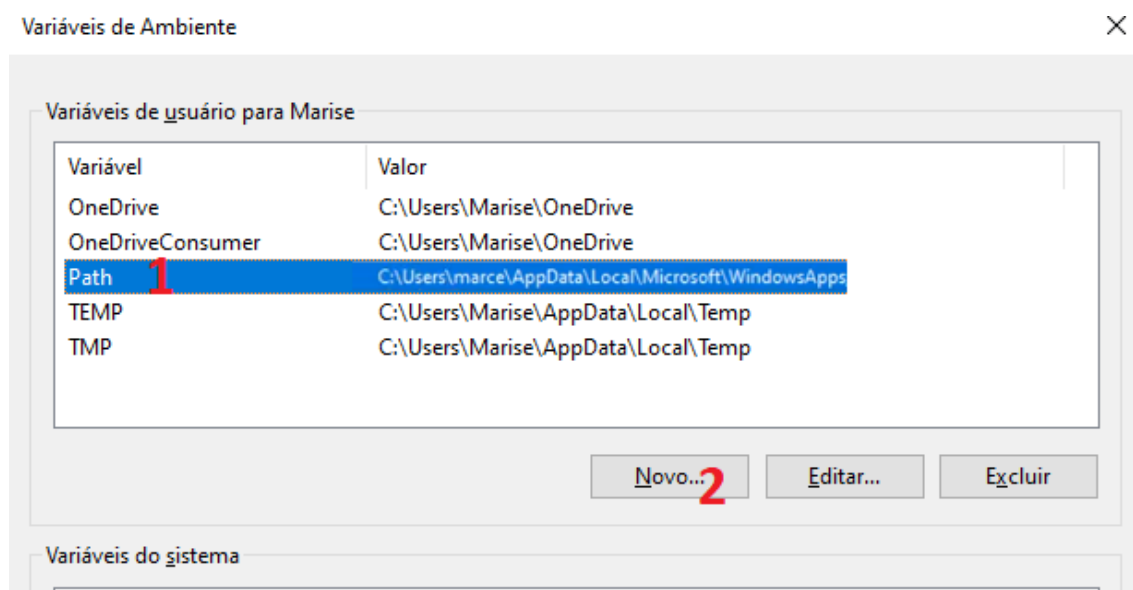
Agora vá em este computador e clique com o botão direito, vá em propriedades:



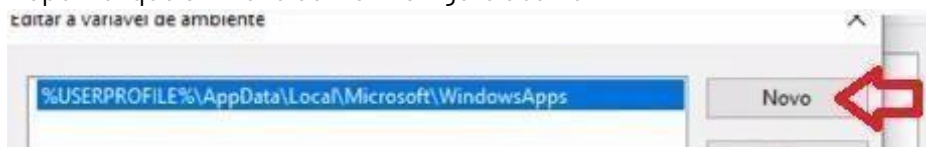
Em propriedades vá em modo avançado:



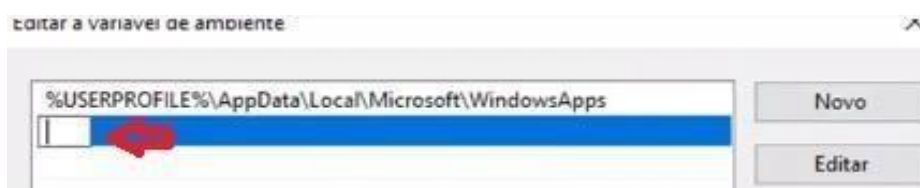
Em variáveis de ambiente, vá em path (1), de pois em novo ( 2), conforme figura seguinte:



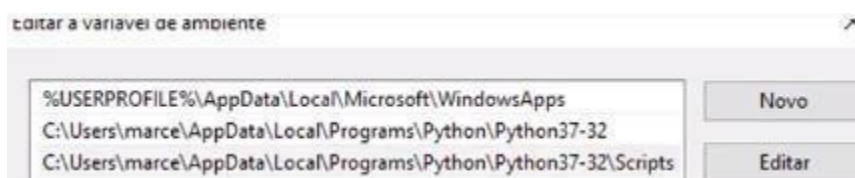
Depois clique em novo conforme figura abaixo:



Depois você vai colar aquele caminho copiado no python executável e vai colar nesse espaço que abriu.



Repita o mesmo procedimento na pasta scripts do python, com o resultado final a seguir:



Dê ok na janela anterior e vá fechando com ok nas janelas seguintes até fechar todas. Pronto agora é só abrir o prompt de comando e chamar o python.

- 1) Salvar o seu código escrito no terminal do INTERPRETADOR, isso é possível?

Diretamente não. Você poderia criar um script e fazer chamada para abrir um arquivo, escrever os arquivos, fechar o arquivo escrito e depois ler o conteúdo do arquivo. Aqui nesta disciplina não faz sentido usar essa estratégia. Ambientes como o IDLE, que é um interpretador mais sofisticados, prefiro chamar assim do que uma ambiente de desenvolvimento e aprendizado integrado, já possuem recursos de escrita e leitura.

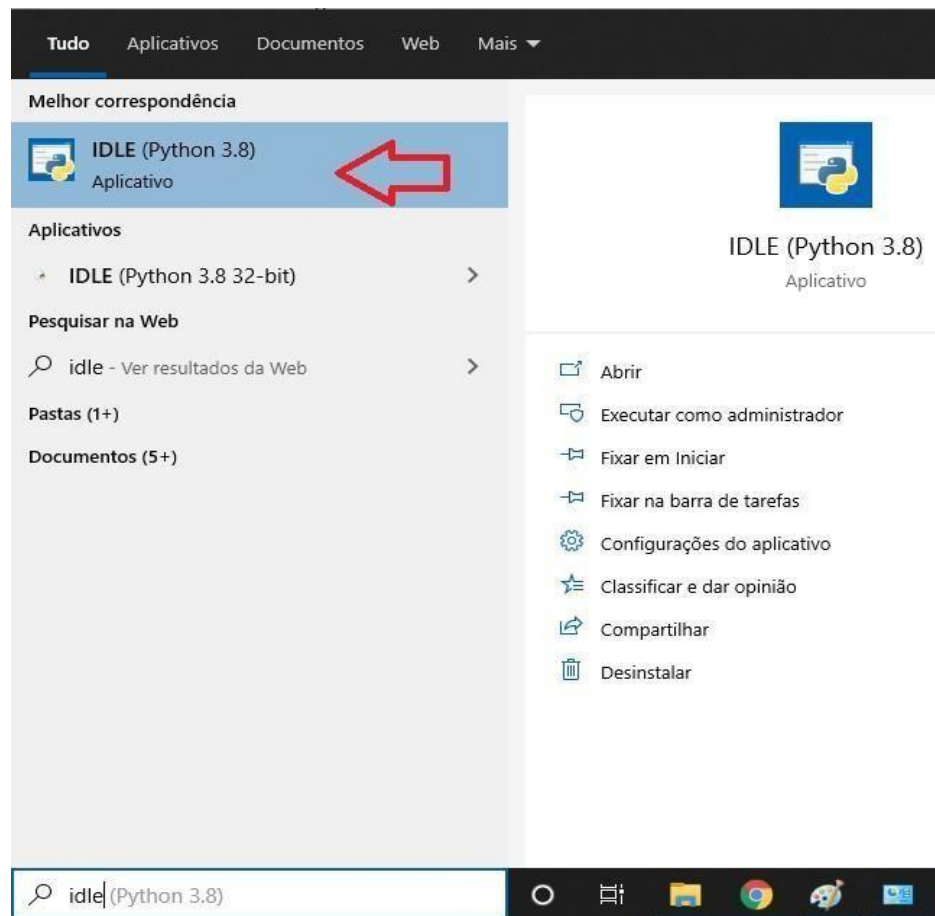
Você pode acessar o IDLE de três maneiras:

- 1) Pelo path via cmd, indo até a pasta do python e chamando a IDLE

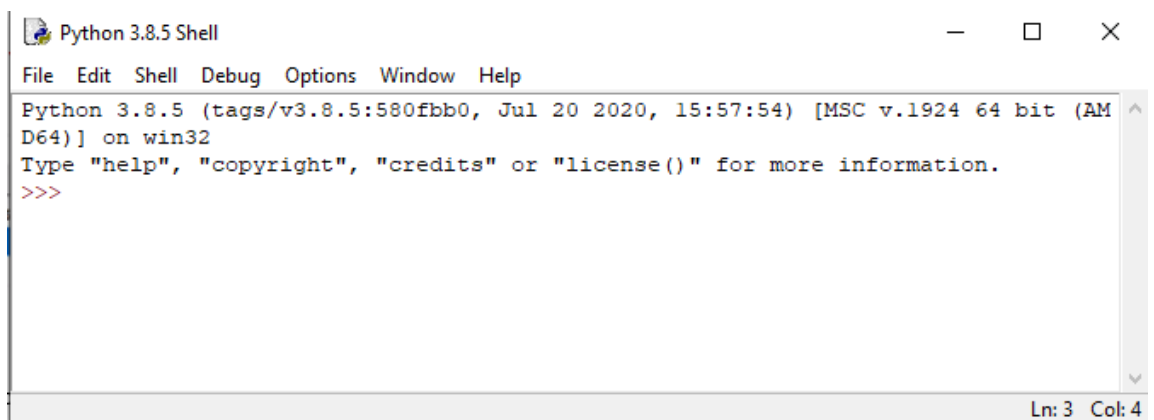
C:\Users\Marise\AppData\Roaming\Microsoft\Windows\Start  
3.8>idle

Menu\Programs\Python

- 2) No menu iniciar digite IDLE, clique que vai abrir o ambiente:



Abra IDLE (Python 3.8)



Observe que há mais recursos, uma barra de ferramentas com várias funcionalidades. Neste caso você poderá salvar seus scripts e códigos.

### Referências Bibliográficas:

PRICE, Ana Maria de Alencar; TOSCANI, Simão Sirineo. Implementação de Linguagens de Programação: compiladores. 3. ed. Porto Alegre: Sagra Luzzatto: Instituto de Informática da UFRGS, 2005.

GERSTING, J. L. Fundamentos matemáticos para a ciência da computação. 5 ed. São Paulo: LCT, 2004.

ALENCAR FILHO, Edgard de. Iniciação à Lógica Matemática. São Paulo: Nobel, 2008.

<https://docs.python.org/3/library/idle.html>

<https://www.python.org/>

<https://repl.it/repls/MonumentalPunctualFlashdrives>

<https://pypi.org/project/CTRL-Z/>