

# MINICURSO PYTHON

Semana da Computação UNIR

# PYTHON

INTRODUÇÃO

MÓDULO 1

MÓDULO 2



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

Quais as vantagens de aprender Phyton?



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

Quais as vantagens de aprender Phyton?



A sintaxe simples e clara do Python torna o processo de aprendizado mais acessível e menos intimidador, permitindo que os novos programadores se concentrem nos conceitos fundamentais sem se perderem em complexidades de sintaxe.

A vasta comunidade e os inúmeros recursos disponíveis, como tutoriais, fóruns e bibliotecas, proporcionam um ambiente de suporte robusto, facilitando a resolução de dúvidas e problemas que possam surgir.

A versatilidade do Python, que é amplamente utilizado em áreas como desenvolvimento web, ciência de dados, inteligência artificial e automação, garante que as habilidades adquiridas sejam aplicáveis a uma ampla gama de oportunidades profissionais, aumentando a empregabilidade e a capacidade de adaptação a diferentes projetos e tecnologias.

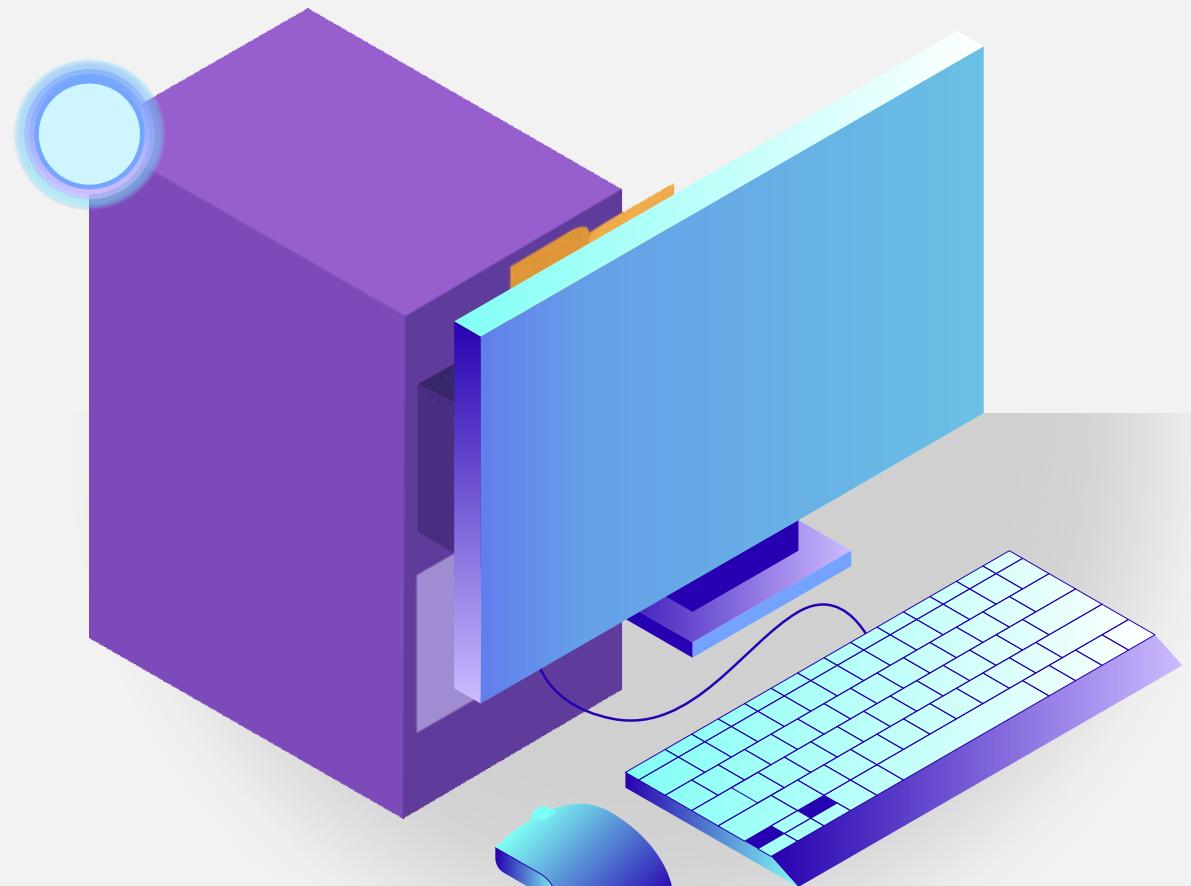


# MINICURSO PYTHON

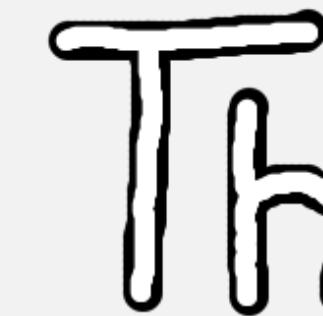
Semana da Computação UNIR

INTRODUÇÃO

Em quais ambientes posso utilizar o  
PYTHON?



Visual Studio Code



**Thonny**  
Python IDE for beginners

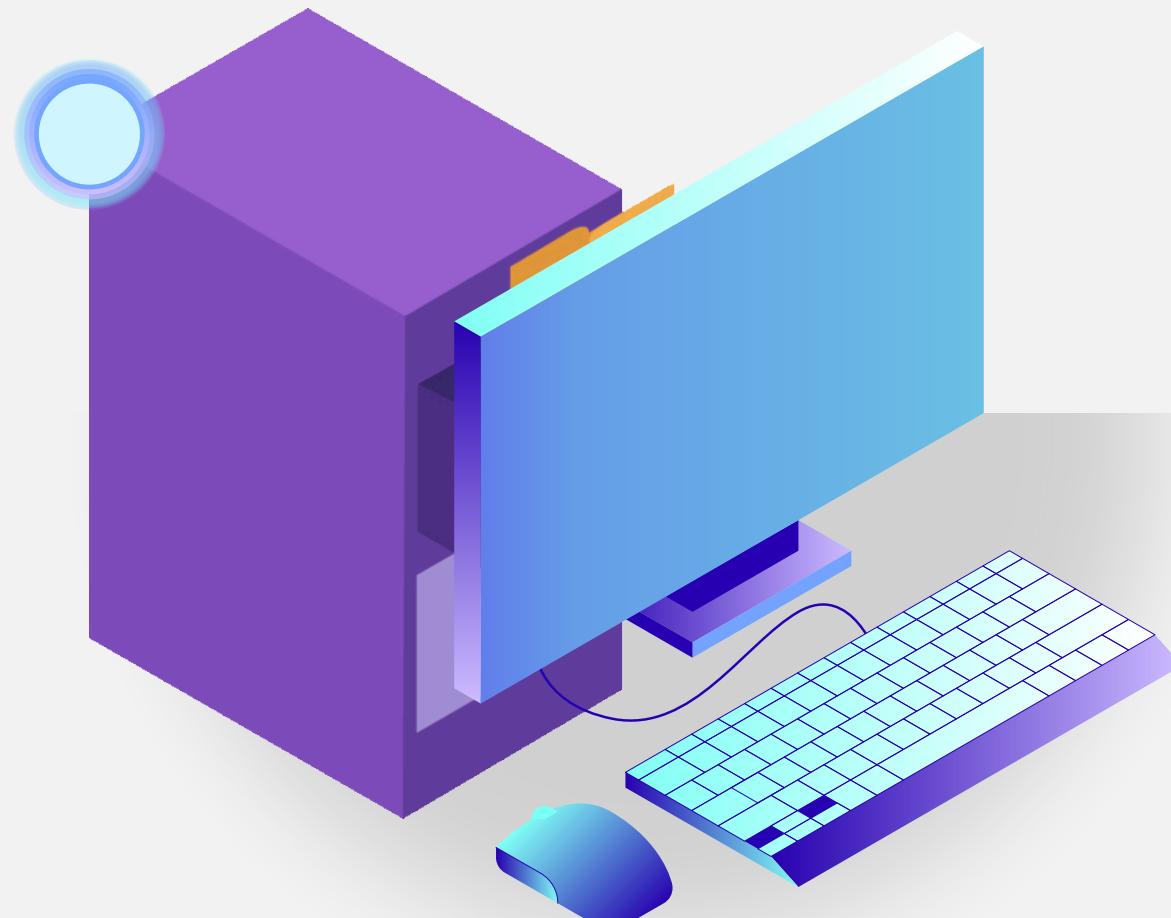


# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Em quais ambientes posso utilizar o PYTHON?



Uma IDE poderosa e completa para Python. Disponível em [jetbrains.com/pycharm](https://www.jetbrains.com/pycharm/)



Ideal para trabalhos em ciência de dados e aprendizado de máquina. Instale usando pip install notebook e execute jupyter notebook no Terminal.



Visual Studio Code

Um editor de código muito popular e extensível. Você pode instalar a extensão Python para suporte completo. Disponível em [code.visualstudio.com](https://code.visualstudio.com)



**THONNY**  
Python IDE for beginners

Um IDE voltado para iniciantes. Disponível em [thonny.org](https://thonny.org).



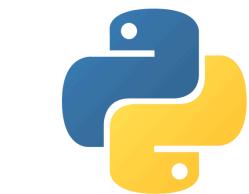
# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO



Qual será nosso  
ambiente de trabalho  
hoje?

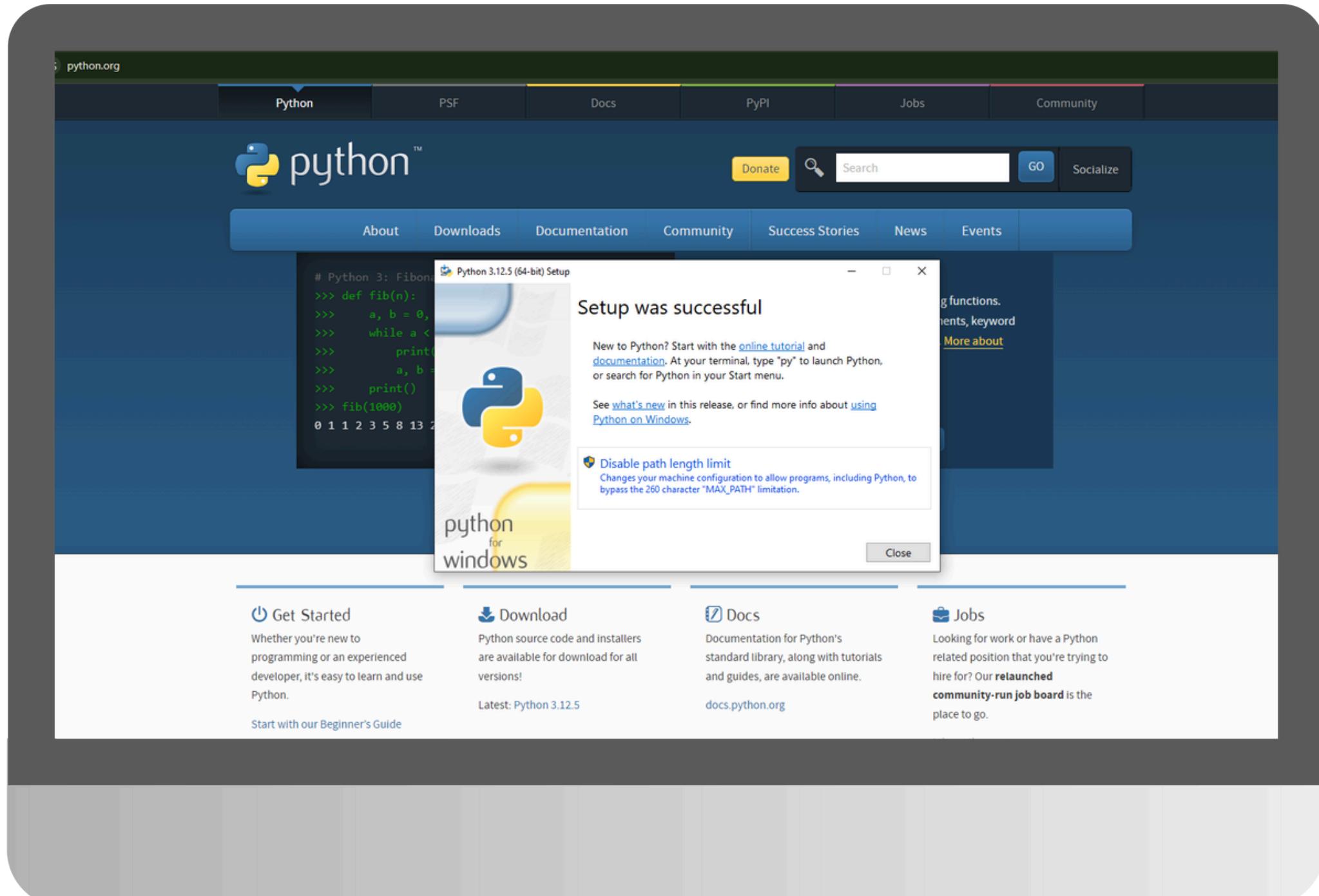


# MINICURSO PYTHON

Semana da Computação UNIR

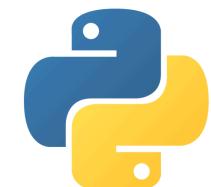
## INTRODUÇÃO

### Preparando as ferramentas



### 1 passo Instalando e verificando

- 1.1: Acessar [python.org](https://python.org) e baixar a versão mais recente.
- 1.2: Abrir o arquivo.exe e adicionar ao PATH.
- 1.3: Verificar instalação no PROMPT com "python --version"

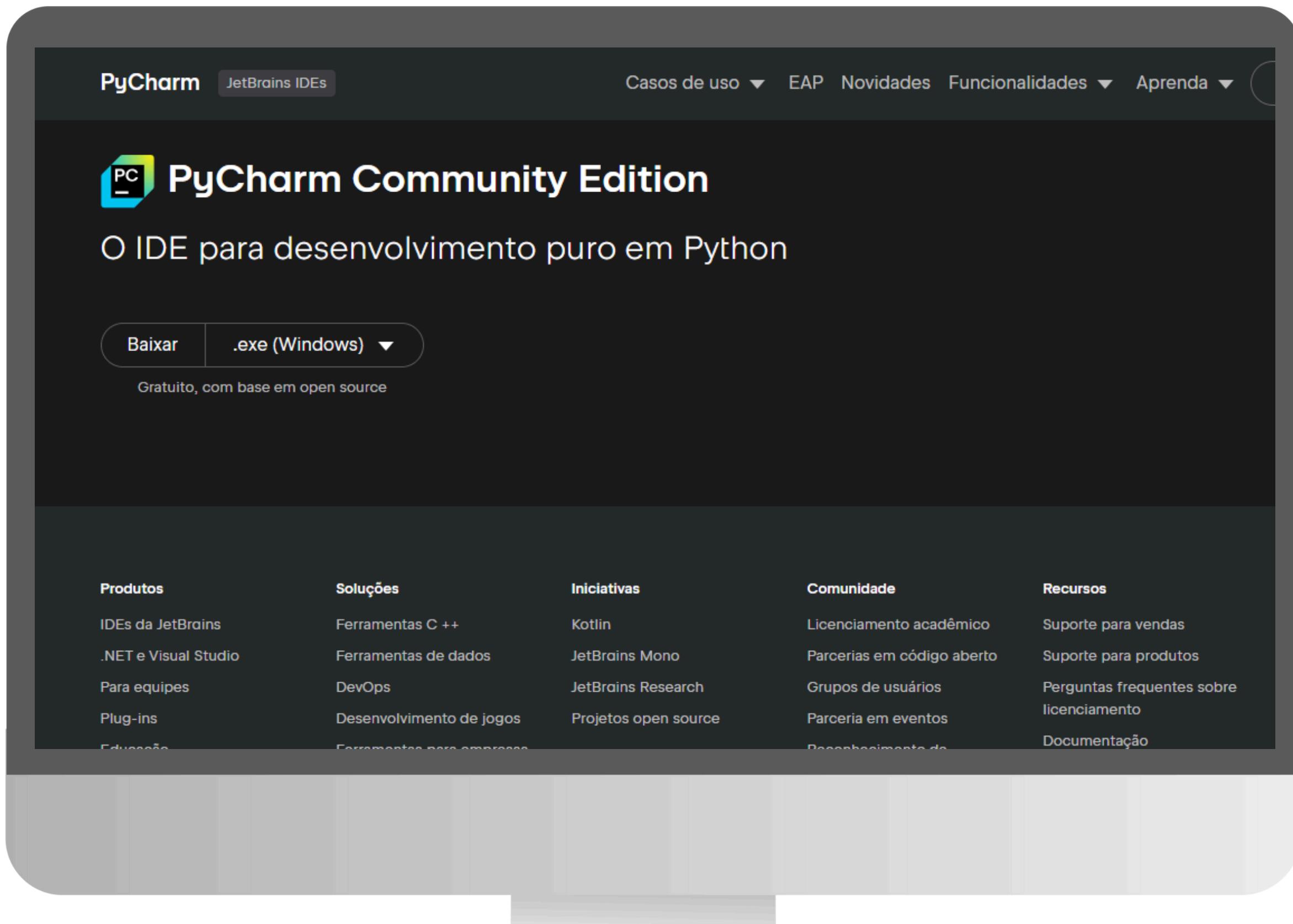


# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

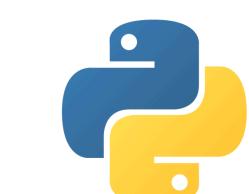
### Preparando as ferramentas



The screenshot shows the official website for PyCharm. At the top, there's a navigation bar with links for 'Casos de uso', 'EAP', 'Novidades', 'Funcionalidades', 'Aprenda', and a search icon. Below the header, the main title is 'PyCharm Community Edition' with a sub-subtitle 'O IDE para desenvolvimento puro em Python'. There are two prominent download buttons: 'Baixar' and '.exe (Windows)'. A note below the buttons says 'Gratuito, com base em open source'. At the bottom of the page, there's a footer with sections for 'Produtos', 'Soluções', 'Iniciativas', 'Comunidade', and 'Recursos', each listing various JetBrains products and services.

2 passo

Vamos instalar o  
PYCHARM agora!



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Sintaxe básica

- Variáveis: criação e tipos de dados.
- Operadores: aritméticos, de comparação e lógicos.
- Comentários.
- Indentação



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO



### Variáveis: criação e tipos de dados

O que são?

Pense em variáveis como rótulos que você atribui a diferentes tipos de dados.

Elas armazenam informações que podem ser usadas mais tarde no seu programa.

Como criar?

Nome da variável = valor

Exemplo:

```
nome = "João"
```



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Sintaxe básica

Tipos de dados:

**int**: números inteiros (1, 2, 3, ...)  
**float**: números com casas decimais (3.14, 2.718)  
**str**: texto (qualquer coisa entre aspas)  
**bool**: valores lógicos (True ou False)

```
Python  
idade = 30 # int  
altura = 1.75 # float  
nome = "Maria" # str  
esta_chovendo = True # bool
```



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Sintaxe básica

Operadores: As Ferramentas para Manipular Dados

- Aritméticos:
  - + (adição), - (subtração), \* (multiplicação), / (divisão), \*\* (potenciação)
- Comparação:
  - == (igual a), != (diferente de), < (menor que), > (maior que), <= (menor ou igual a), >= (maior ou igual a)
- Lógicos:
  - and (e), or (ou), not (não)



Python

```
x = 10  
y = 5  
print(x + y) # Imprime 15  
print(x > y) # Imprime True
```



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Sintaxe básica

#### Comentários: Explique o Seu Código

- Para quê?
  - Tornar o código mais fácil de entender para você e outras pessoas.
  - Documentar o que cada parte do código faz.
- Como fazer?
  - `#` para comentários de uma linha
  - `"""` para comentários de múltiplas linhas



#### Python

```
# Calcula a área de um círculo
raio = 5
area = 3.14159 * raio**2
print("A área do círculo é:", area)
```



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Estruturas Condicionais

Estruturas Condicionais: Tomando Decisões no Seu Código

- if, else, elif:
  - if: Se uma condição for verdadeira, execute um bloco de código.
  - else: Se a condição do if for falsa, execute este bloco.
  - elif: Se a condição do if for falsa, verifique outra condição.
- Operadores de comparação:
  - == (igual a)
  - != (diferente de)
  - < (menor que)
  - > (maior que)
  - <= (menor ou igual a)
  - >= (maior ou igual a)

```
■ ● ▲  
Python  
idade = 17  
  
if idade >= 18:  
    print("Você pode dirigir.")  
else:  
    print("Você não pode dirigir.")
```



# MINICURSO PYTHON

Semana da Computação UNIR

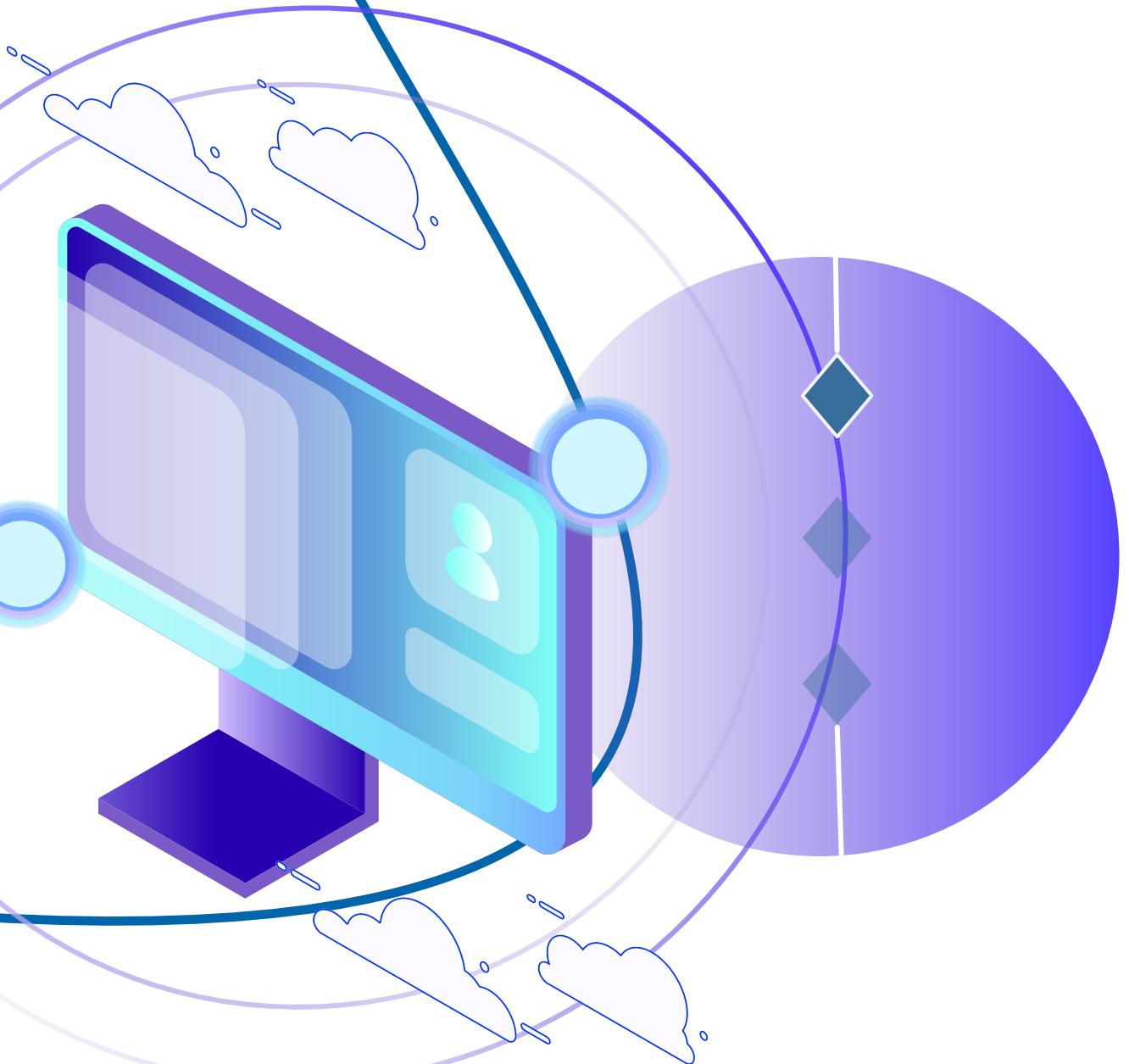
## INTRODUÇÃO

### Estruturas Condicionais

E se quisermos verificar várias condições?

- **elif:** Permite adicionar mais condições.

```
■ ● ▲  
Python  
#codigo para verificar notas de provas  
nota = 7  
  
if nota >= 9:  
    print("Aprovado!")  
elif nota >= 7:  
    print("Aprovado.")  
else:  
    print("Reprovado.")
```



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Estruturas Condicionais

Os operadores de comparação são essenciais para criar essas condições:

- if idade  $\geq$  18: Verifica se a idade é maior ou igual a 18.
- if nota  $\geq$  9: Verifica se a nota é maior ou igual a 9.

As estruturas condicionais são como bifurcações em um caminho. Elas permitem que seu programa tome decisões com base em diferentes condições.



# MINICURSO PYTHON

Semana da Computação UNIR

## INTRODUÇÃO

### Estruturas Condicionais

Além dos operadores de comparação, existem outros tipos de operadores que, embora não sejam de comparação direta como os listados acima, são usados para comparar ou verificar condições:

```
Python
a = [1, 2, 3]
b = a
a is b # Retorna Verdadeiro!
Ou também, podemos utilizar:
a is not b # Retorna Falso!
```

```
Python
# Calcula a área de um círculo
raio = 5
area = 3.14159 * raio**2
print("A área do círculo é:", area)
```



# MINICURSO PYTHON

Semana da Computação UNIR

## Estruturas Condicionais

### Atividade desafio:



Crie um programa que peça ao usuário sua idade e verifique se ele pode votar (idade mínima de 16 anos). Se ele puder votar, informe também se ele é obrigado a votar (entre 18 e 70 anos).

```
Python  
#Desenvolva seu código :)
```

## INTRODUÇÃO





### Estruturas de repetição



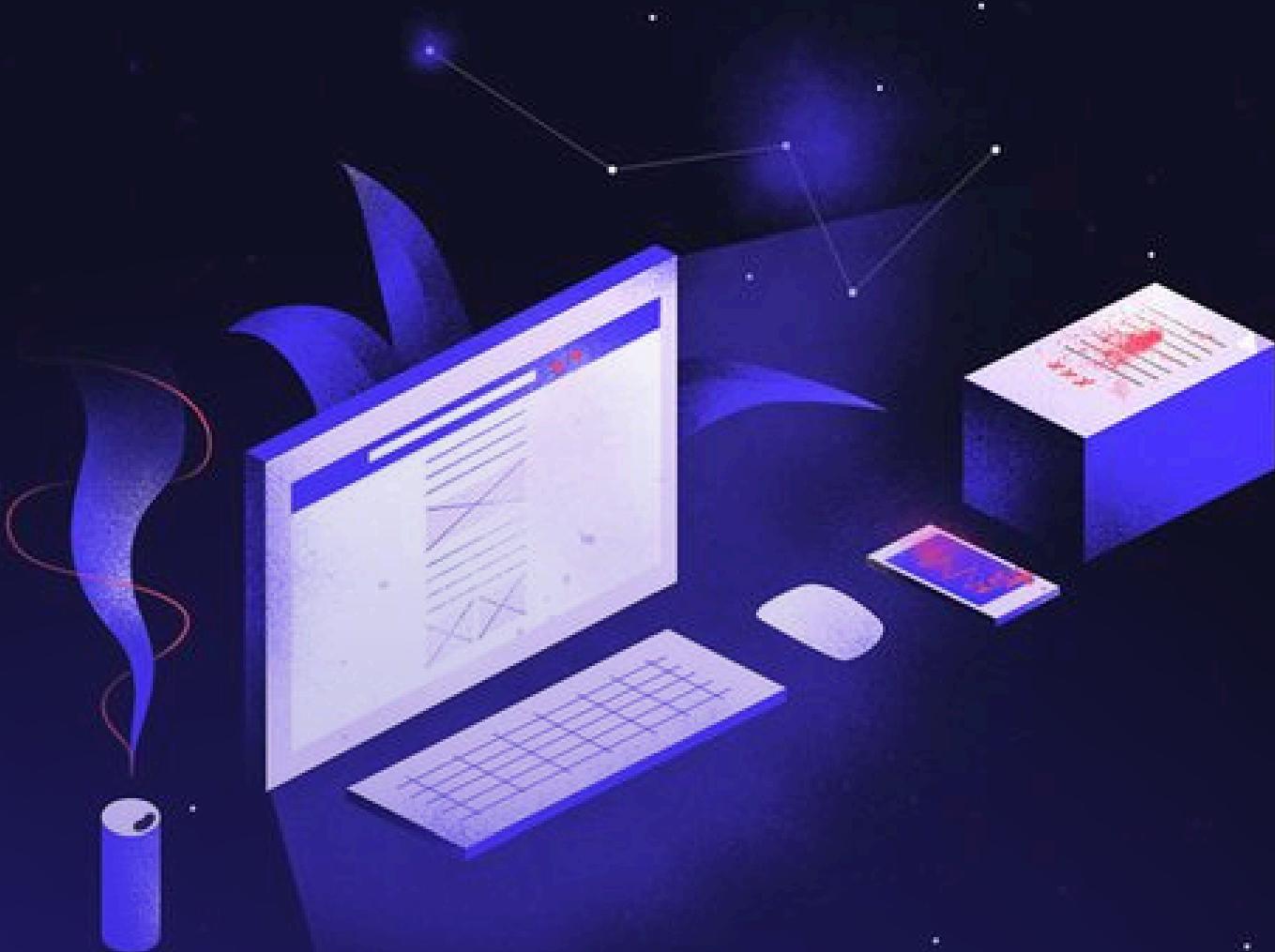
#### O que são estruturas de repetição?

Estruturas de repetição, ou loops, são ferramentas poderosas em programação que permitem que um bloco de código seja executado repetidamente enquanto uma determinada condição for verdadeira. **Elas são essenciais para automatizar tarefas repetitivas e realizar cálculos complexos.**

#### Por que usar loops?

- Automatização: Evitam a repetição manual de código.
- Processamento de dados: Permitem percorrer listas, strings e outras estruturas de dados.
- Cálculos iterativos: Realizam cálculos que dependem de resultados anteriores.





### Estruturas de repetição



Quando usar for e while?

- **for:**
  - Utilizado quando se sabe o número exato de iterações.
  - Ideal para percorrer sequências (listas, tuplas, strings).
  -
- **while:**
  - Utilizado quando a condição de parada não é conhecida a priori.
  - Ideal para repetir um bloco de código enquanto uma condição for verdadeira.

Como funcionam?

```
Python
for i in range(5):
    print(i)
```

range(5) cria uma sequência de números de 0 a 4. A cada iteração, i assume um valor da sequência. O bloco de código dentro do for é executado 5 vezes.





while:



```
Python  
contador = 0  
while contador < 5:  
    print(contador)  
    contador += 1
```

1. A condição `contador < 5` é verificada antes de cada iteração.
2. O bloco de código é executado enquanto a condição for verdadeira.
3. O contador é incrementado em cada iteração para evitar um loop infinito.



# MINICURSO PYTHON

Semana da Computação UNIR

## Estruturas de repetição

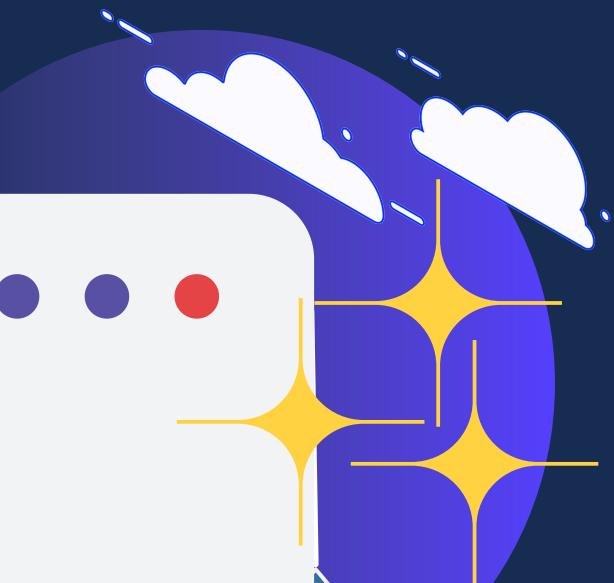
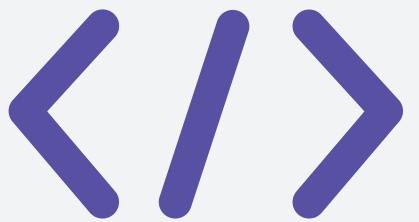
### Atividade desafio:



Crie um programa que peça ao usuário um número e imprima a tabuada desse número.

```
Python  
#Desenvolva seu código :)
```

## INTRODUÇÃO

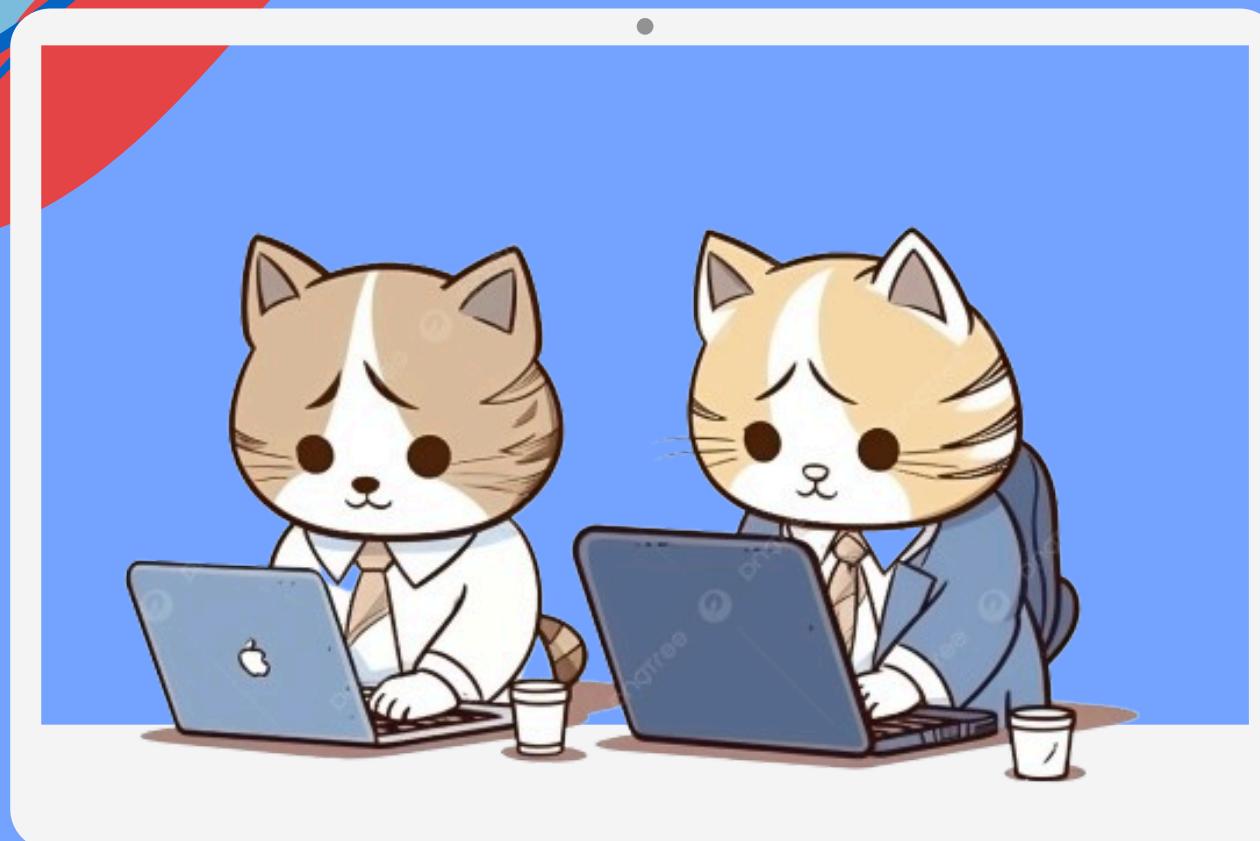




## Listas e Vetores



Listas e Vetores são Estruturas de dados, que são fundamentais para armazenar, organizar e manipular grandes quantidades de dados.

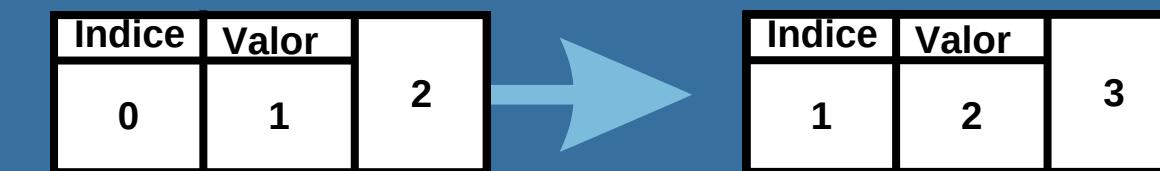
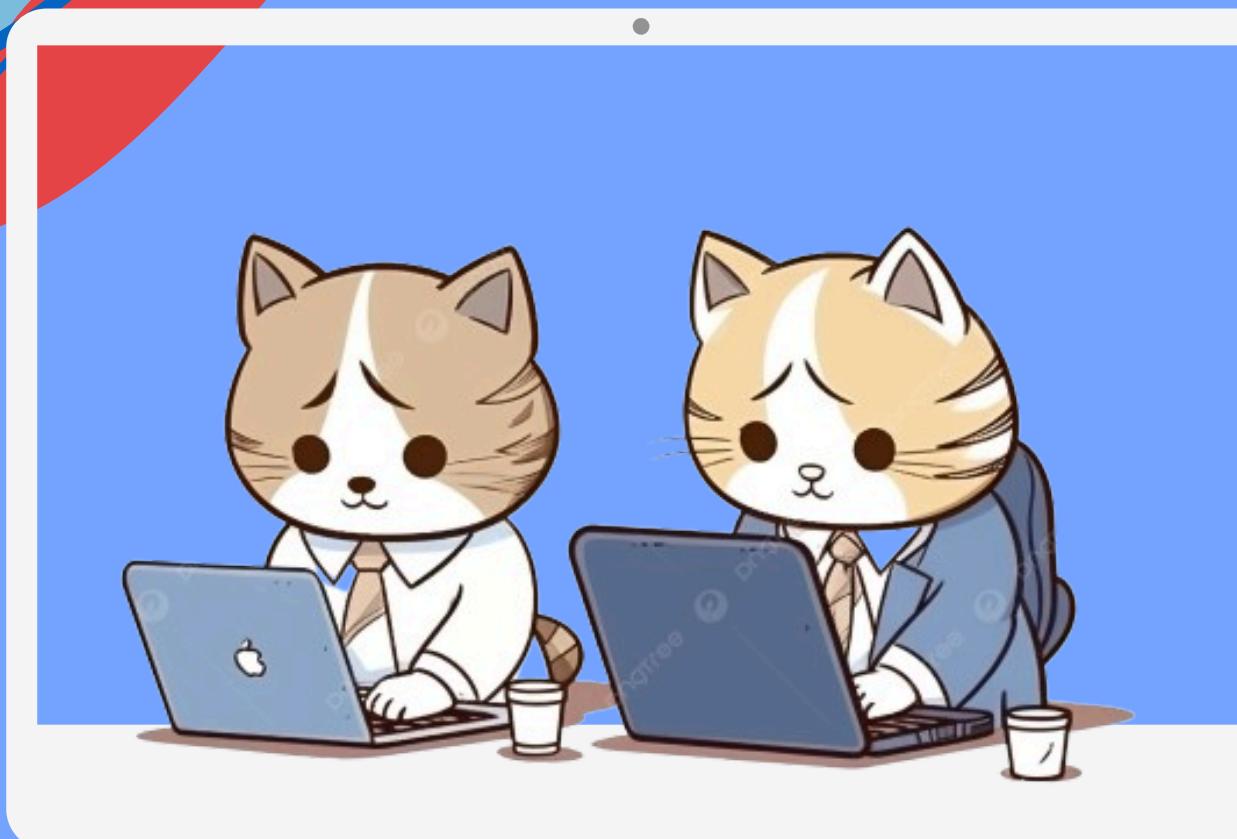


As listas em Python permitem armazenar múltiplos valores em uma única variável, oferecendo suporte a diferentes tipos de dados, como números, strings, ou até mesmo ambos!

```
Python
Lista Vazia = []
Lista com Números = [3, 2, 1]
Lista Strings = ["a", "b", "c"]
Lista Mista = [2, "python", False]
```



Você pode acessar elementos da lista utilizando **índices**!



Segue exemplos de como acessar elementos no Python:



```
Python  
a = [10, 20, 30]  
print(a[0]) #Saída: 10.  
print(a[1]) #Saída: 20.  
print(a[-1]) #Saída: 30.
```

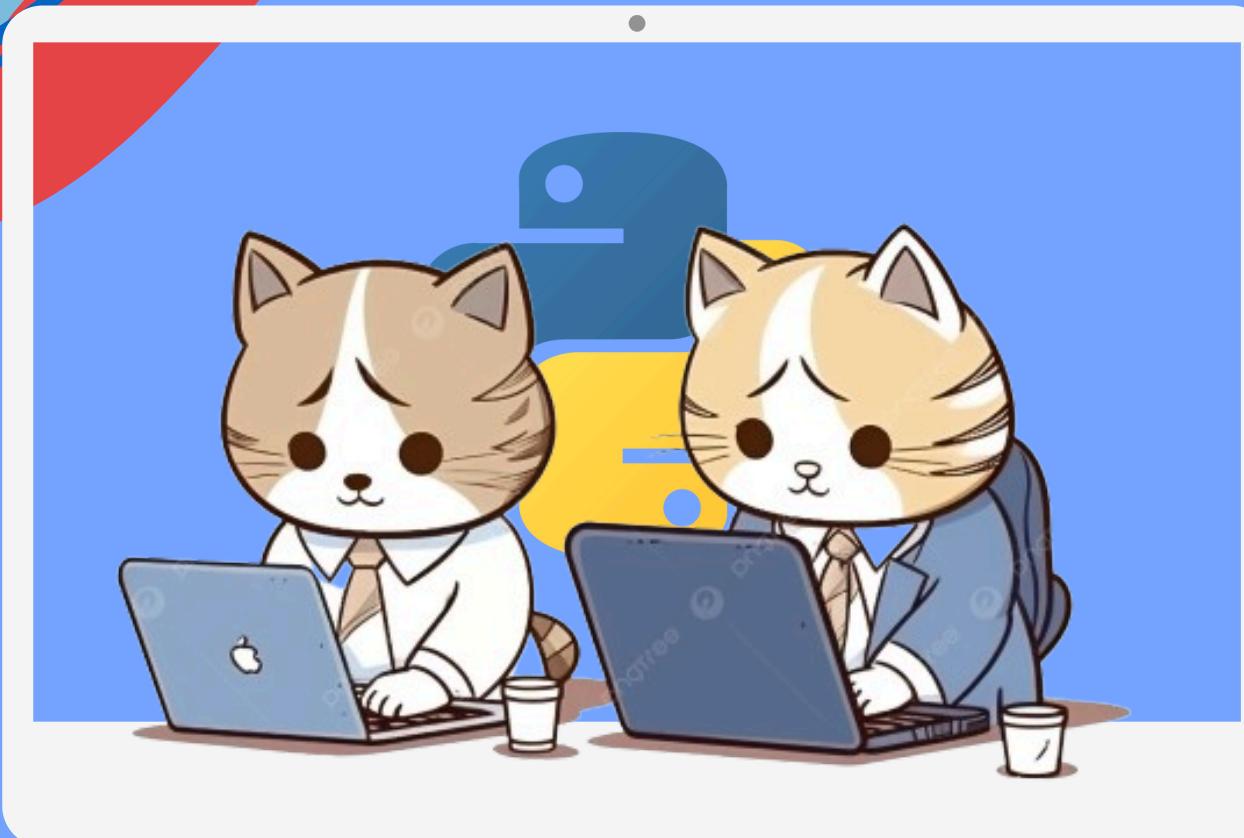


A inserção em listas também pode ser realizada utilizando funções!

- `append(valor)` - Adiciona um item ao final da Lista.
- `insert(índice,valor)` - Insere um item em uma posição.

Como é realizada a exclusão de elementos?

- `pop(valor)` - Remove um item ao final da Lista.
- `remove(índice,valor)` - Remove item específico.
- `clear()` - Limpa toda a Lista.





Para o uso situacional podemos **ordenar** nossas listas conforme nossa vontade com as seguintes funções:

- `sort()` - Ordena a Lista em Ordem Crescente.
- `reverse()` - Reverte a Ordem da Lista.
- `lista.sort(reverse=True)` - Ordena em Ordem Decrescente

Vamos ter uma visualização no código!



Python

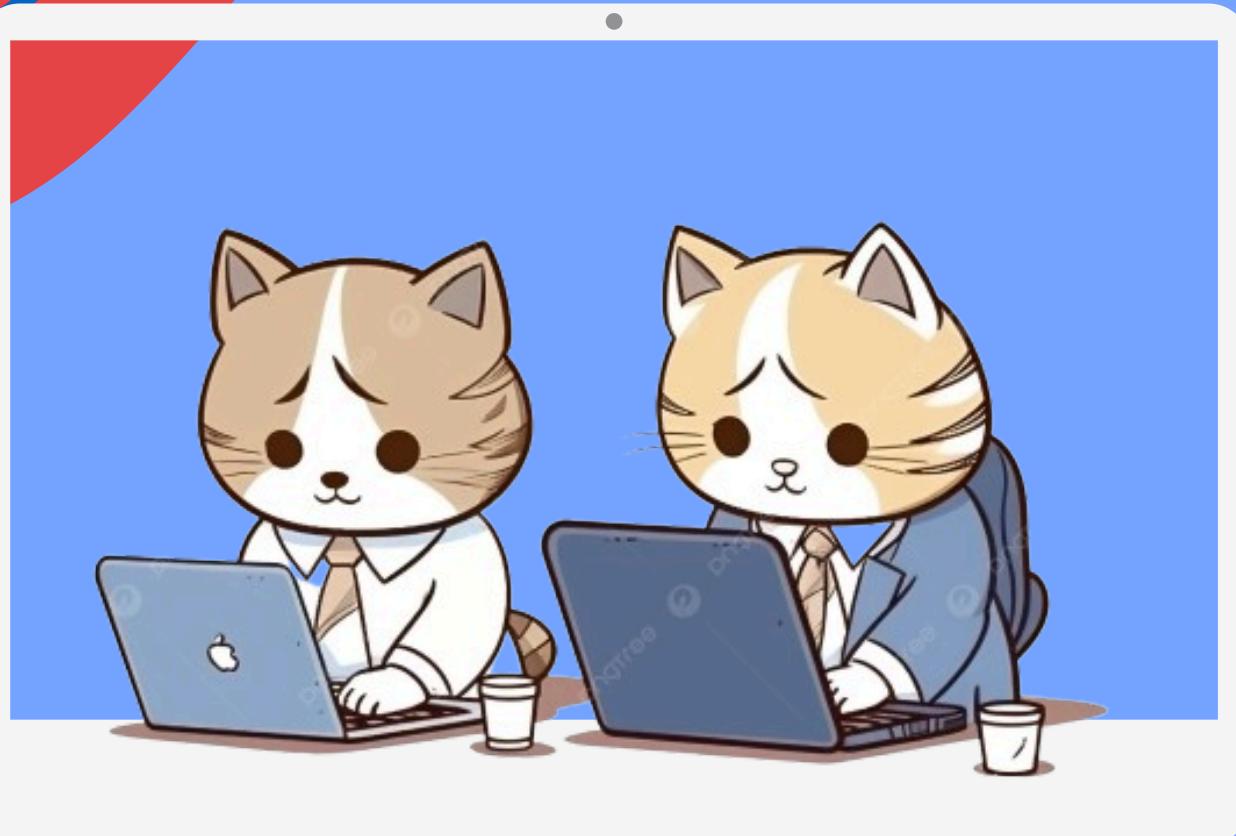
```
Lista = [40,10,20,30]
```

```
Lista.sort()
```

```
print(Lista) # Saída: [10, 20, 30, 40]
```

```
Lista.reverse()
```

```
print(Lista) # Saída: [40, 30, 20, 10]
```





### Algumas funções que são úteis para a Operação com Listas:

- len() - Retorna o número de elementos em uma lista.
- max() - Retorna o maior valor da lista.
- min() - Retorna o menor valor da lista.
- sum() - Soma todos os elementos da lista.

Podemos utilizar técnicas de simplificação de Condicionais utilizando o operador “IN”



```
lista = [10, 20, 30, 40]
print(20 in lista) # Saída: True
print(50 in lista) # Saída: False
```



# MINICURSO PYTHON

Semana da Computação UNIR

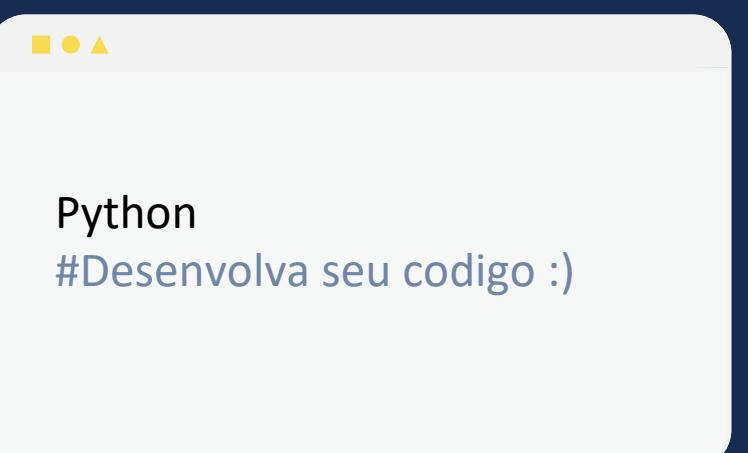
## Listas e Vetores

Atividade desafio: 

- Crie um programa que solicite ao usuário o nome de 5 produtos, suas quantidades e preços. Armazene essas informações em três listas separadas: uma para os nomes, outra para as quantidades e outra para os preços.

Extra:

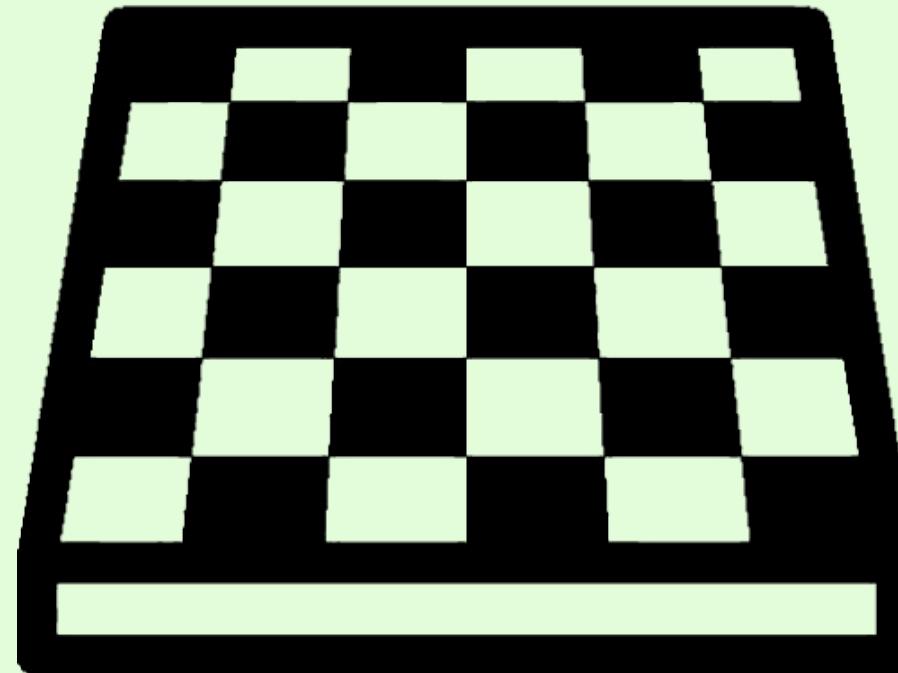
- Exiba os dados inseridos de forma organizada, mostrando o nome, quantidade e preço de cada produto.



### Introdução á Matrizes

#### O que são Matrizes?

Matrizes em Python são coleções bidimensionais de dados, organizadas em linhas e colunas, semelhantes a tabelas, ou tabuleiros.



Elas podem ser representadas usando listas aninhadas (listas dentro de listas)



### Matrizes em Python

Em formato de código, podemos ver abaixo como criar uma matriz simples usando listas aninhadas:

```
■ ● ▲  
Python  
matriz = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 9]  
]  
# Ou também :  
matriz = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]
```

Também podemos visualizar como fazemos a como acessar os índices da matriz:

```
■ ● ▲  
Python  
matriz = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]  
print(matriz[0][1]) # Saída 2.  
  
matriz[1][2] = 10 # Troca 5 por 10.
```



### Matrizes em Python

Considerando a Matriz:

```
[1, 2, 3]  
[4, 5, 6]  
[7, 8, 9]
```

Podemos percorrer a matriz, utilizando os laços de repetição como o **FOR** e o **WHILE** !

```
# Percorrendo a matriz linha por linha  
for i in range(len(matriz)):  
    # Percorrendo cada elemento da linha  
    for j in range(len(matriz[i])):  
        print(f"Elemento na posição [{i}][{j}] é {matriz[i][j]}")
```



### Operação entre Matrizes

Operações entre matrizes envolvem somar, subtrair ou multiplicar matrizes, e que as operações devem respeitar o tamanho das matrizes (dimensões compatíveis).

#### Operação de Soma:

```
matriz1 = [[1, 2], [3, 4]]  
matriz2 = [[5, 6], [7, 8]]  
  
resultado = [[matriz1[i][j] + matriz2[i][j] for j in  
range(len(matriz1[0]))] for i in range(len(matriz1))]  
  
print(resultado) # Saída: [[6, 8], [10, 12]]
```

#### Operação de Multiplicação:

```
matriz1 = [[1, 2], [3, 4]]  
matriz2 = [[5, 6], [7, 8]]  
escalar = 3  
  
resultado = [[escalar * matriz1[i][j] for j in  
range(len(matriz1[0]))] for i in range(len(matriz1))]  
  
print(resultado) # Saída: [[3, 6], [9, 12]]
```



# MINICURSO PYTHON

Semana da Computação UNIR

## Matrizes

Atividade desafio:



- Faça um código que dada uma matriz NxN, imprima toda a Diagonal Principal.
- EXTRA:**
- Imprima toda a diagonal Secundária!



Python

#Desenvolva seu código :)

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & & & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

diagonal principal  $i = j$



# MINICURSO PYTHON

Semana da Computação UNIR

MÓDULO 2



O que são Funções?

Funções são blocos de código reutilizáveis que realizam uma tarefa específica. Elas ajudam a organizar o código, tornando-o mais legível e modular.

A screenshot of a code editor window titled 'Python'. It contains the following code:

```
def nome_da_funcao(parametro1, parametro2):
    # Código a ser executado
    return resultado
```



# MINICURSO PYTHON

Semana da Computação UNIR

MÓDULO 2



Abaixo temos alguns exemplos de Funções!

```
def saudacao(nome):
    return f"Olá, {nome}!"

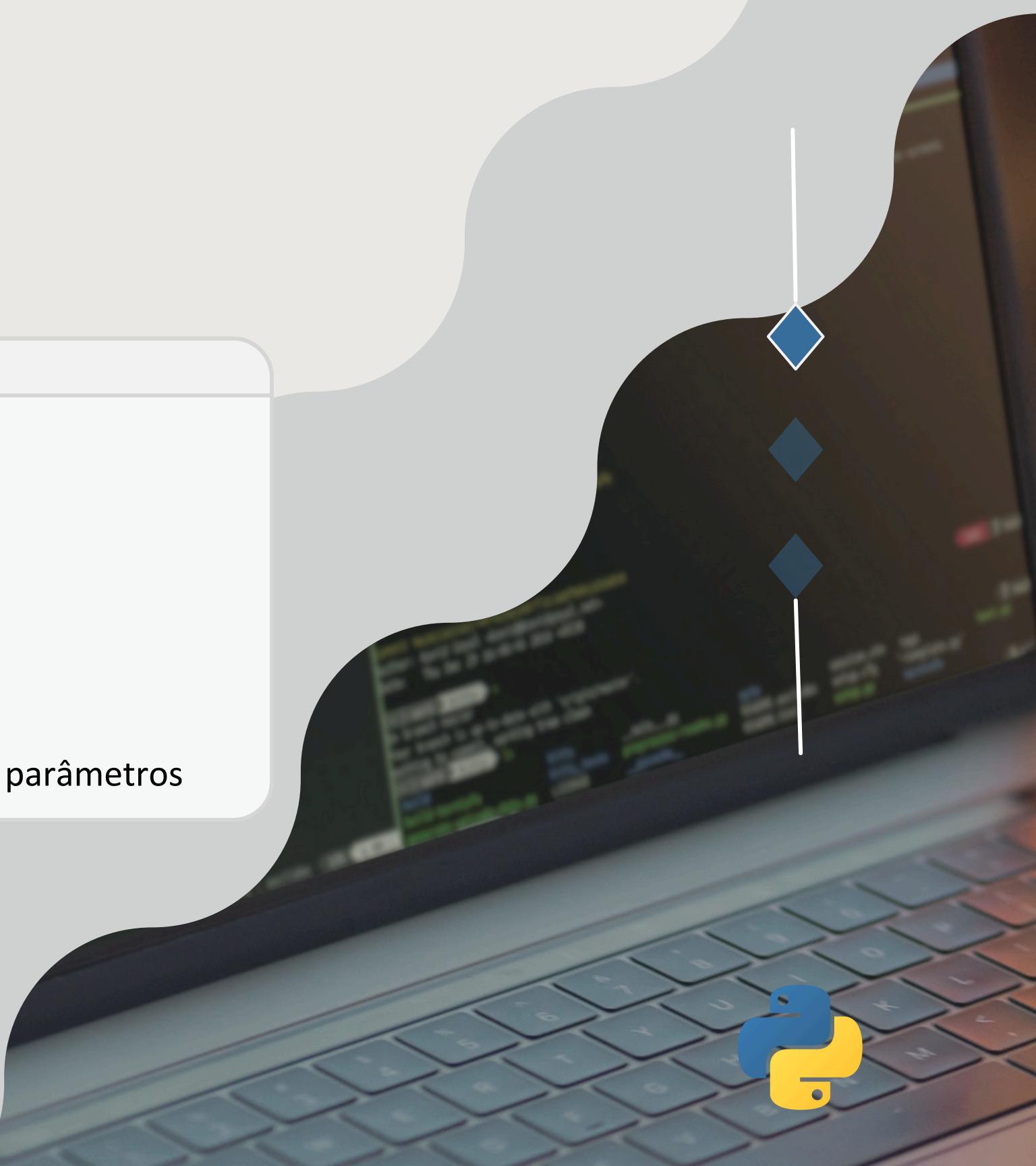
print(saudacao("Maria")) # Saída: Olá, Maria!
#exemplo de código simples
```



```
Python

def soma(a, b):
    return a + b
resultado = soma(5, 3)
print(resultado) # Saída: 8

#exemplo de função com vários parâmetros
```





O que são funções recursivas?

Uma função recursiva é uma função que chama a si mesma dentro de sua própria definição.

```
def fibonacci(n):
    if n == 0: # Caso base 1: Fibonacci de 0 é 0
        return 0
    elif n == 1: # Caso base 2: Fibonacci de 1 é 1
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
```

Funções recursivas são úteis quando o problema possui uma natureza repetitiva ou pode ser decomposto em versões menores de si mesmo.



# MINICURSO PYTHON

Semana da Computação UNIR

MÓDULO 2



expressões lambda são uma maneira de criar funções anônimas e podem ser usadas em funções como map(), filter() e sorted(). permitindo escrever funções em uma única linha.

A screenshot of a Python code editor window titled "Python". The code in the editor is:

```
f = lambda x: x * 2
print(f(5)) # Saída: 10
```



# MINICURSO PYTHON

Semana da Computação UNIR

## Funções

### Atividade desafio:



- Escreva uma função em Python que, dado um número n, retorne o fatorial desse número. Sinta-se livre para resolver o problema utilizando qualquer abordagem ou método (recursivo, iterativo, bibliotecas, etc.).

```
Python
#Desenvolva seu código :)
```



# OBRIGADO!

Dúvidas?

(69) 9 99233-3203  
(69) 9 9957-3103

Jáder Louis  
Mariana Feitoza

Publique uma foto do seu projeto Python e nos marque!

@mari\_poong  
@jader.louis  
@daccunir



Siga nosso departamento no Instagram:

