Clarify

Python

O que é Python:

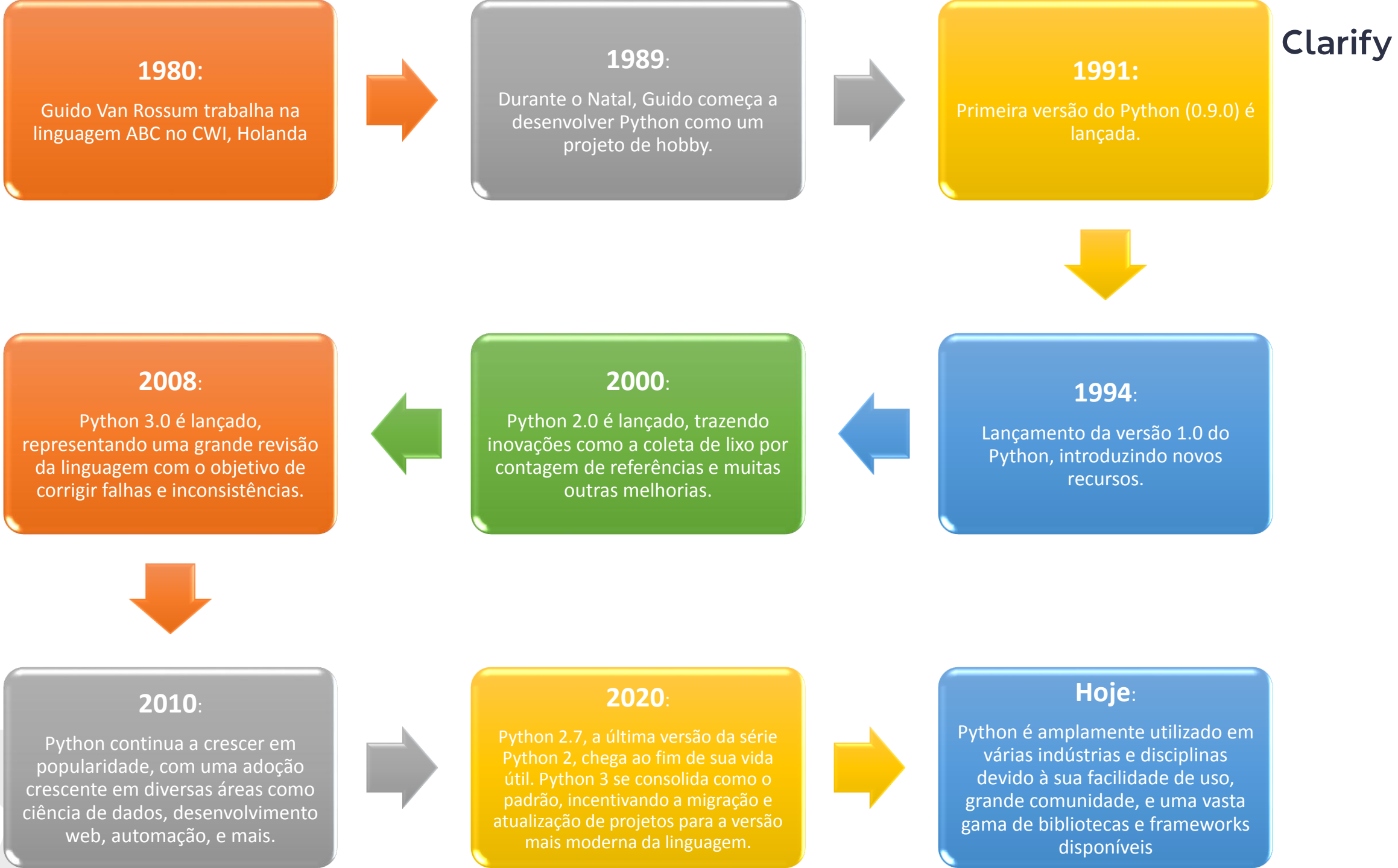
Python é uma linguagem de programação altamente popular e versátil, destacando-se por sua simplicidade e legibilidade.

Com sua sintaxe clara e uma vasta coleção de bibliotecas, Python capacita os desenvolvedores a criar uma ampla gama de aplicações, desde programas simples até projetos complexos.

História do Python

Python foi criado por **Guido van Rossum** e sua primeira versão foi lançada em 1991.

Inspirado por linguagens como ABC, C, e o Unix, Guido queria desenvolver uma linguagem que fosse fácil de aprender e usar, mantendo a capacidade de ser poderosa e flexível.


1980:
Guido Van Rossum trabalha na linguagem ABC no CWI, Holanda
1989:
Durante o Natal, Guido começa a desenvolver Python como um projeto de hobby.
1991:
Primeira versão do Python (0.9.0) é lançada.
1994:
Lançamento da versão 1.0 do Python, introduzindo novos recursos.
2000:
Python 2.0 é lançado, trazendo inovações como a coleta de lixo por contagem de referências e muitas outras melhorias.
2008:
Python 3.0 é lançado, representando uma grande revisão da linguagem com o objetivo de corrigir falhas e inconsistências.
2010:
Python continua a crescer em popularidade, com uma adoção crescente em diversas áreas como ciência de dados, desenvolvimento web, automação, e mais.
2020:
Python 2.7, a última versão da série Python 2, chega ao fim de sua vida útil. Python 3 se consolida como o padrão, incentivando a migração e atualização de projetos para a versão mais moderna da linguagem.
Hoje:
Python é amplamente utilizado em várias indústrias e disciplinas devido à sua facilidade de uso, grande comunidade, e uma vasta gama de bibliotecas e frameworks disponíveis

O que é possível fazer com Python

•Desenvolvimento Web

- Criação de websites dinâmicos e interativos
- Desenvolvimento de APIs
- Automação de testes e tarefas de manutenção

•Automação de Tarefas

- Scripts para automação de processos repetitivos
- Manipulação e organização de arquivos
- Envio automatizado de e-mails

•Análise de Dados

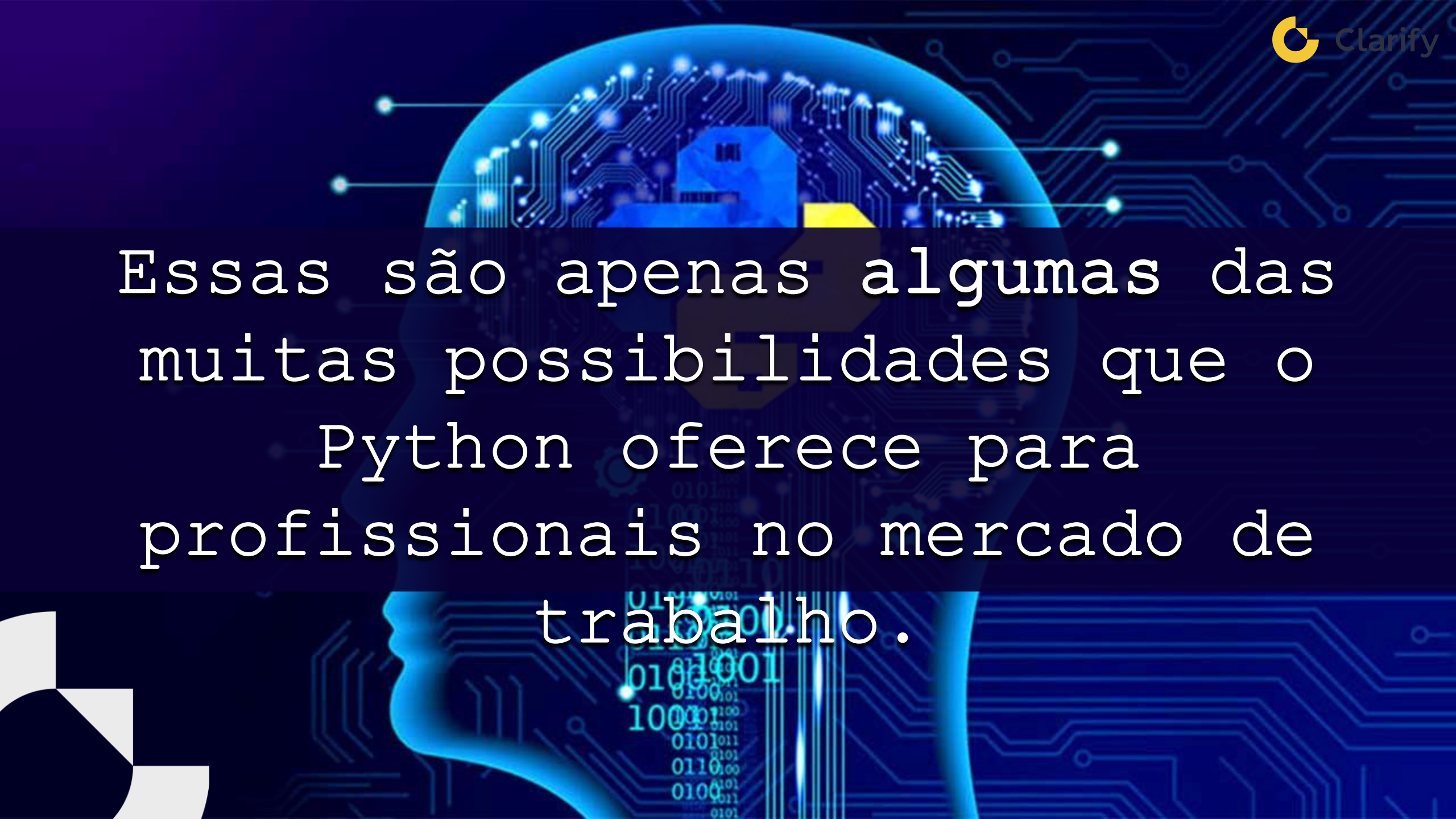
- Processamento e limpeza de grandes volumes de dados
- Criação de relatórios e visualizações interativas
- Análise estatística e preditiva

Desenvolvimento de Software

- Criação de aplicações desktop
- Desenvolvimento de scripts para ferramentas internas
- Prototipagem rápida de novas ideias

Entretenimento e Mídia

- Desenvolvimento de jogos e aplicativos de mídia
- Análise de dados de mídias sociais
- Automação de tarefas de produção de conteúdo


Clarify
Essas são apenas algumas das muitas possibilidades que o Python oferece para profissionais no mercado de trabalho.

Quem está usando Python hoje?

Desenvolvedores de Software	Ciência de Dados e Análise:	IA e Machine Learning:	Educação	Grandes Empresas de Tecnologia	VOCÊ
Python é amplamente utilizado no desenvolvimento web, desenvolvimento de jogos, automação de tarefas, entre outros.	Python é a linguagem dominante em ciência de dados, com bibliotecas como Pandas, NumPy, Matplotlib e TensorFlow sendo muito populares.	Python é amplamente adotado em projetos de IA e aprendizado de máquina devido às suas bibliotecas poderosas, como TensorFlow, PyTorch e Scikit-learn.	Muitas instituições educacionais ensinam Python devido à sua simplicidade e aplicabilidade em várias áreas.	Empresas como Google, Facebook, Amazon, Microsoft e muitas outras utilizam Python em diferentes capacidades, desde desenvolvimento de software até automação e análise de dados.	Que busca se aperfeiçoar, mudar de carreira ou ingressar nesse fantástico mundo de programação

Introdução a lógica de Programação

O que é

A lógica de programação é um conceito fundamental para quem deseja iniciar no mundo da programação, especialmente utilizando a linguagem Python. Ela consiste em uma forma estruturada de pensar e resolver problemas, utilizando sequências de instruções lógicas para alcançar um determinado objetivo.

Importância da lógica de Programação

Para iniciantes em Python, compreender a lógica de programação é de extrema importância, pois é a base para o desenvolvimento de algoritmos e a criação de programas funcionais. Ao entender como a lógica funciona, os iniciantes poderão escrever códigos mais eficientes, organizados e facilmente compreensíveis.

Variáveis

- **Variável:**

  Uma variável é um tipo de armazenamento de dados em memória, que possui o conteúdo variável durante a execução de um algoritmo ou programa. Uma variável pode assumir vários valores diferentes ao longo da execução do programa, mas, em um determinado momento, possui apenas um valor.

- **Constante:**

  Uma constante é um tipo de armazenamento de dados em memória, que possui um valor fixo e imutável, durante a execução de um algoritmo ou programa.

Variáveis

As Variáveis devem possuir um tipo de dado, que pode ser:

1. Numérico;
2. Literal;
3. Lógico.

- Toda a variável possui um Identificador, que representa o nome escolhido para rotular a variável.
- Após recapitularmos estes pontos, vamos compreender como o “portugol” trabalha com as variáveis:

Tipos de dados em “portugol”

Tipo de dado	Descrição	Exemplos
inteiro	Armazena qualquer número inteiro, negativo, nulo ou positivo.	-2, -1, 0, 1, 2
real	Armazena qualquer número real, negativo, nulo ou positivo.	2.5, 3.1
caracter	Armazena apenas um caractere alfanumérico.	'a', 'b', 'c'
cadeia	Armazena uma cadeia de caracteres alfanuméricos de qualquer tamanho.	"casa", "lógica", "123"
logico	Tipo especial, que armazena apenas os valores VERDADEIRO e FALSO.	

Entradas e Saídas

Entrada e Saída de dados:

Existem basicamente duas instruções principais em algoritmos que são:

- **Leia** (entrada) e o **Escreva** (saída).

A instrução escreva é utilizada para mostrar informações na tela do computador, ou seja, é um Comando de Saída de Dados em tela.

A instrução leia é utilizada quando se deseja obter informações do usuário por meio do teclado, ou seja, é um Comando de Entrada de Dados.

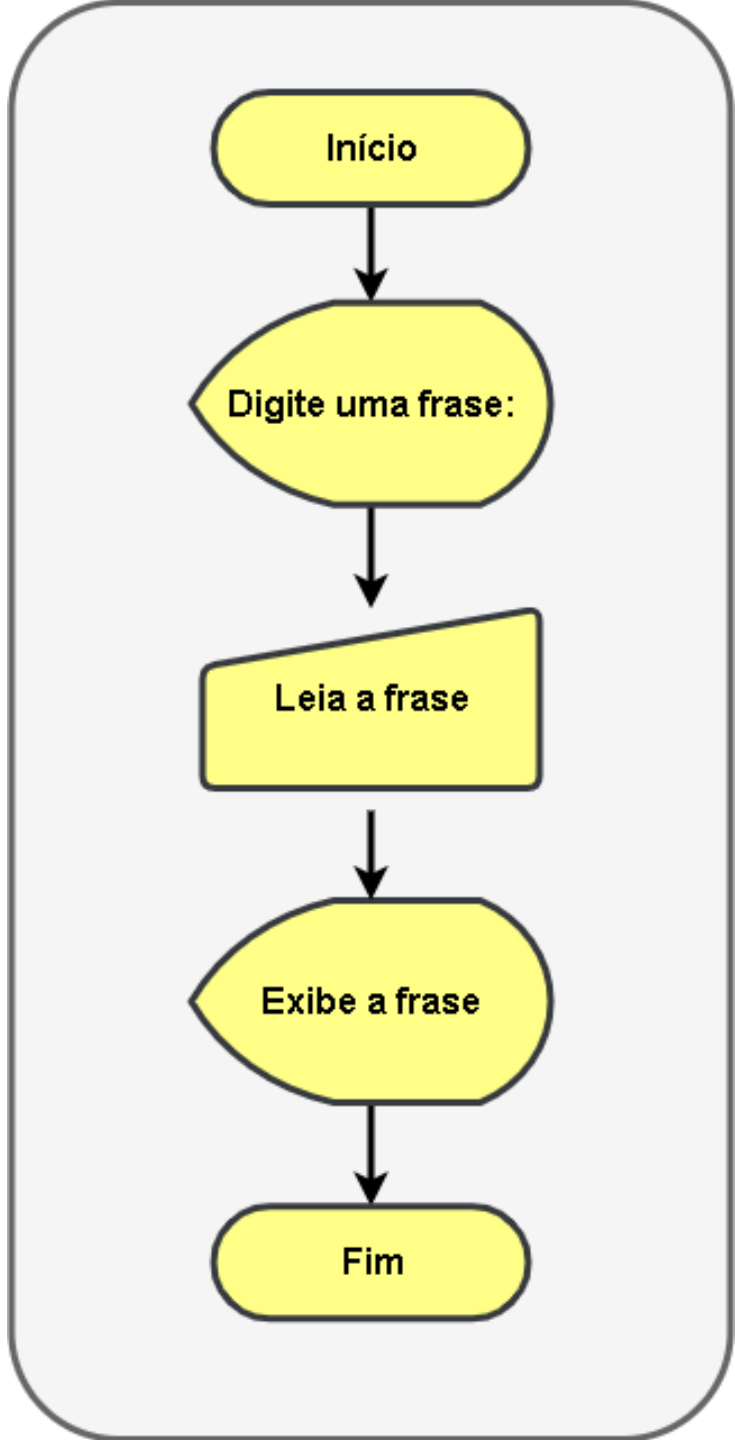
Código no Portugol:

```
programa

{
	funcao inicio()
	{
		cadeia frase
	escreva("\nDigite uma frase: ")


		leia(frase)

	escreva("\nO valor armazenado na
Variável frase é: " + frase)
			}
		}
```



Operadores

Operadores são símbolos que representam atribuições, cálculos e ordem dos dados.

Os Operadores são elementos funcionais, que atuam (Processamento) sobre os operandos (Entrada de dados), e produzem um determinado resultado (Saída de dados).

Por exemplo, a expressão 3 + 2 relaciona dois operandos (os números 3 e 2) por meio do operador (+) que representa a operação de adição.

Em relação ao tipo de dados, os Operadores são classificados como:

- Operadores Aritméticos;
- Operadores de Atribuição;
- Operadores Relacionais.
- Operadores Lógicos.

Operadores Aritméticos

Os Operadores Aritméticos são um conjunto de símbolos que representam as operações básicas da matemática (Soma, Subtração e etc). Esses operadores somente poderão ser utilizados entre variáveis com os tipos de dados Numéricos Inteiros e/ou Numéricos Reais

Operador	Operação
+	**Soma**
-	**Subtração**
*	**Multiplicação**
/	**Divisão**
%	**Módulo ou Resto**

Operadores de Atribuição

Os Operadores de Atribuição têm como função retornar um valor atribuído de acordo com a operação indicada. A operação é feita entre os dois operandos, sendo atribuído o resultado ao primeiro.

Operador	Descrição
=	Atribuição simples
+=	Atribuição com soma
-=	Atribuição com subtração
*=	Atribuição com multiplicação
/=	Atribuição com divisão

Operadores Relacionais

Os Operadores Relacionais são utilizados para comparar valores entre variáveis e expressões do mesmo tipo e criar declarações condicionais. Esses operadores são usados com o intuito de criar expressões do tipo verdadeiro (TRUE) ou falso (FALSE), fundamentais para as declarações condicionais. O retorno da comparação é sempre um valor do tipo Lógico.

Operador	Descrição
>	Maior do que
>=	Maior do que ou igual
<	Menor do que
<=	Menor do que ou igual
==	Igual
!=	Diferente

Operadores lógicos

Os Operadores Lógicos são utilizados para realizar comparações e validações, criando expressões condicionais complexas.

Assim como podemos comparar objetos e lugares, dentro de um software também podemos comparar dados. Podemos, por exemplo, verificar se o e-mail e a senha digitados são iguais aos cadastrados no sistema, se a idade informada pelo usuário é maior ou igual a 18 anos, se um campo do formulário foi preenchido ou está vazio, entre outras tantas comparações que precisamos fazer para o bom funcionamento de uma aplicação.

Esses operadores são usados com o intuito de criar expressões do tipo verdadeiro (TRUE) ou falso (FALSE), mas o seu funcionamento é um pouquinho mais complexo.

Operadores lógicos

Operação Lógica	Operador	Descrição
Conjunção	**e**	Valida se dois dados são verdadeiros. É escrito uma letra **e** entre duas proposições.
Disjunção	**ou**	Válida se um dos dois dados apresentados são verdadeiros, ou se os dois são verdadeiros. É escrito com a palavra **ou** entre as duas proposições.
Negação	**nao**	Inverte o valor, o que é verdadeiro passa a ser falso, o que é falso passa a ser verdadeiro. É a negação de uma sentença. É escrito com a palavra **não** antes da proposição.

Estruturas Condicionais

As estruturas condicionais em lógica de programação, voltadas para Python, são fundamentais para controlar o fluxo de execução do código com base em condições específicas. Aqui estão os principais elementos que você precisa conhecer:

- Estrutura básica do condicional if
- Condicionais if, elif e else
- Operadores de comparação

ESTRUTURA BÁSICA DO CONDICIONAL IF

Clarify

```python
idade = 18

if idade >= 18:
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")

print("Isso será sempre impresso.")
```

Neste exemplo:

O código dentro do if será executado SE a variável idade for maior ou igual a 18, caso contrário, o código no else será executado.

CONDICIONAIS IF, ELIF E ELSE

Clarify

```python
nota = 85

if nota >= 90:
    print("Aprovado com A")
elif nota >= 80:
    print("Aprovado com B")
elif nota >= 70:
    print("Aprovado com C")
else:
    print("Reprovado")
```

Neste exemplo:

Se nota for 85,
a primeira condição (nota >= 90) é falsa
então verifica a próxima condição (elif nota >= 80), que é verdadeira.

Portanto, "Aprovado com B" será impresso.

O uso de elif permite avaliar várias condições em sequência, enquanto else captura qualquer outra situação que não foi coberta pelas condições anteriores.

OPERADORES DE COMPARAÇÃO

```python
a = 5
b = 10

print("a =", a)
print("b =", b)

# Igual a
if a == b:
    print("a é igual a b")
else:
    print("a não é igual a b")

# Diferente de
if a != b:
    print("a é diferente de b")
else:
    print("a não é diferente de b")
```

```python
# Maior que
if a > b:
    print("a é maior que b")
else:
    print("a não é maior que b")

# Menor que
if a < b:
    print("a é menor que b")
else:
    print("a não é menor que b")

# Maior ou igual a
if a >= b:
    print("a é maior ou igual a b")
else:
    print("a não é maior ou igual a b")
```

```python
# Menor ou igual a
if a <= b:
    print("a é menor ou igual a b")
else:
    print("a não é menor ou igual a b")
```

Saída esperada:

```
a = 5
b = 10
a não é igual a b
a é diferente de b
a não é maior que b
a é menor que b
a não é maior ou igual a b
a é menor ou igual a b
```

Estrutura de Repetição

As estruturas de repetição permitem executar um bloco de código repetidamente enquanto uma condição específica for verdadeira ou para iterar sobre elementos em uma sequência. Aqui estão os principais tipos de estruturas de repetição em Python:

While

O laço while executa um bloco de código repetidamente enquanto uma condição especificada for verdadeira.

For

O laço for é utilizado para iterar sobre uma sequência (como uma lista, tupla, dicionário, conjunto, etc.) ou um iterável.

Instruções break e continuebreak:

Encerra imediatamente o laço mais próximo.continue: Pula para a próxima iteração do laço mais próximo.

Compreensão de listaUma técnica avançada para criar listas de forma concisa usando loops for:

Clarify

clarify.com.br

Av. Paulista, 1439 - 2° andar - Bela Vista - São Paulo - SP
(11) 94358-7033 | 3675-0033
contato@clarify.com.br