Engenharia de Software
Computacional thinking with Python
Aula 01

Prof. Dr. Francisco Elânio

Agenda da aula

- Revisão sobre alguns conceitos do semestre passado
- Ementa para o segundo semestre
- Bibliografia
- Avaliação
- Resolução de exercícios

Conteúdo – 2° semestre

- Subalgoritmos
- Funções | Procedimentos
- Parâmetros: Reais e formais | default | args
- Estrutura de dados unidimensional
- Estruturas de dados do tipo: vetor (dados homogêneos) | lista (list) | tupla (tuple)
- Estrutura de dados homogênea bidimensional tipo matriz (dados homogêneos)
- Busca e ordenação em vetores / listas

Conteúdo – 2° semestre

- Conjuntos (set)
- Dicionários
- Tabelas de memória (listas x dicionários)
- Arquivos
- Manipulação de arquivos texto
- Manipulação de arquivos CSV

Avaliação

A média final para aprovação deve ser maior ou igual a 6,0 pontos. Essa nota será resultante do novo sistema de avaliação da FIAP, assim composto:

Média 1º semestre:

- Challenge Sprint (2 atividades) e
- Checkpoints (2 a 3), representando 40% da nota semestral;
- Uma Global Solution, representando 60% da nota do semestre.

Obs 1: Essa nota representará 40% da média anual.

Média 2º semestre:

- Challenge Sprint (2 atividades) e
- Checkpoints (2 a 3), representando 40% da nota semestral;
- Uma Global Solution, representando 60% da nota do semestre.

Obs 2: Essa nota representará 60% da média anual.

Bibliografia

BÁSICA

- ASCENCIO, A. F. G, CAMPOS, E. A. V. Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java, 2ª Edição, São Paulo: Pearson 2007.
- FURGERI, Sérgio. Introdução à Programação em Python. São Paulo: Editora Senac, 2021.
- MENEZES, Nilo. Introdução à Programação em Python. São Paulo: Novatec, 2019
- SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madson. Algoritmos. São Paulo: Pearson, 2004.

COMPLEMENTAR

- BORGES, Luiz Eduardo. Python para Desenvolvedores, 2ª Edição. Rio de Janeiro: Novatec, 2010.
- MATTHES, Eric. Curso Intensivo de Python: Uma introdução prática e baseada em projetos à programação. São Paulo: Novatec, 2016.
- SWEIGART, Al. Automatize Tarefas Maçantes com Python: Programação prática para verdadeiros iniciantes. São Paulo: Novatec, 2015.

Introdução à linguagem de programação em Python

Linguagem de programação Python

Python é uma linguagem de programação de alto nível, dinâmica, interpretada, modular, multiplataforma, orientada e administrada pela Python Software Foundation.

Possui sintaxe simples e oferece suporte a módulos e pacotes.

Autor da linguagem de programação Python

Guido Van Rossum

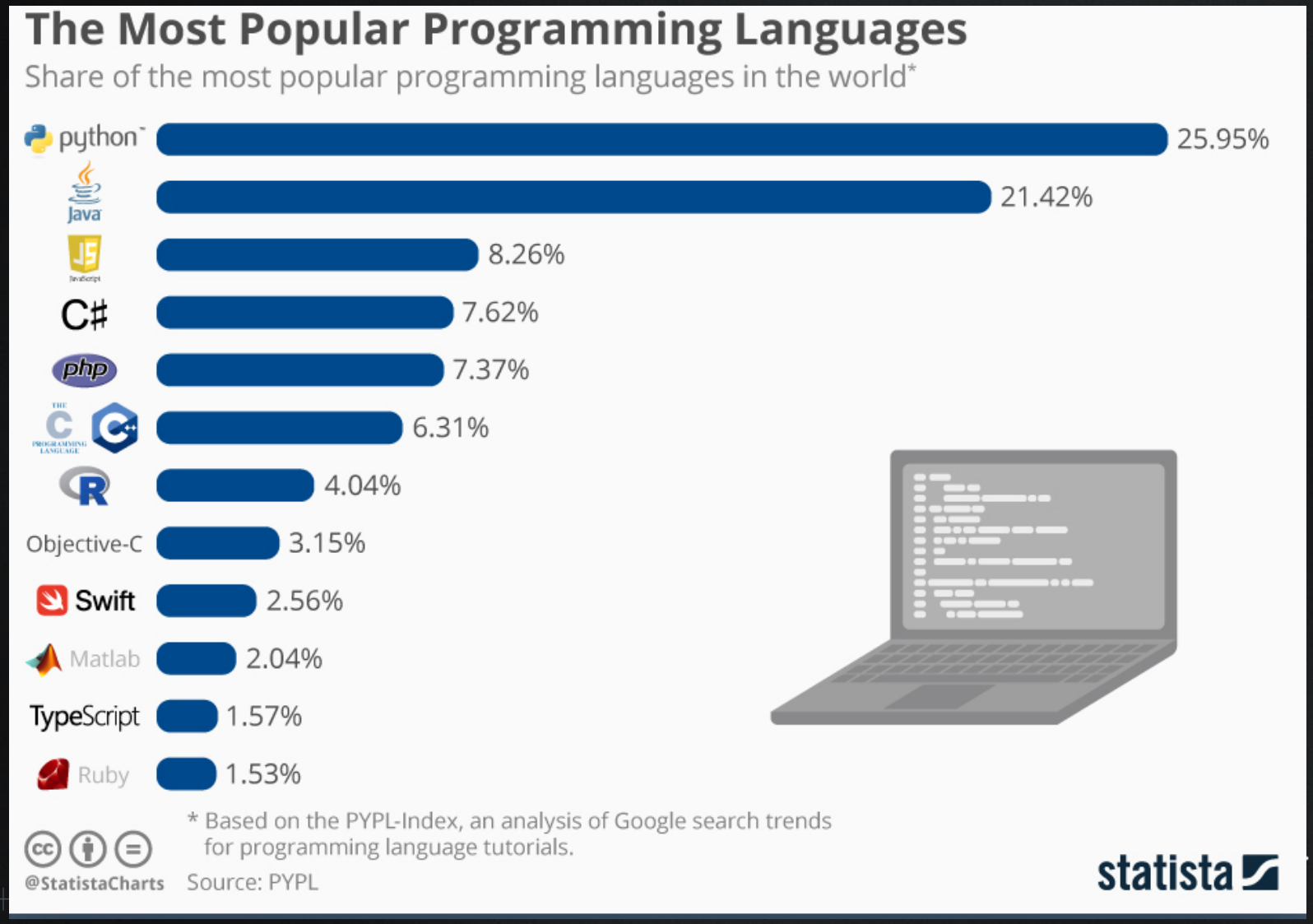
Nascimento em 31 de janeiro de 1956 (67 anos)

Haarlem, Países Baixos

Prêmio Anual Software Livre (2001)

Programador de computador na Holanda. Iniciou o seu projeto em 1989 no *Centrum Wiskunde & Informatica* (CWI).

Linguagens mais utilizadas

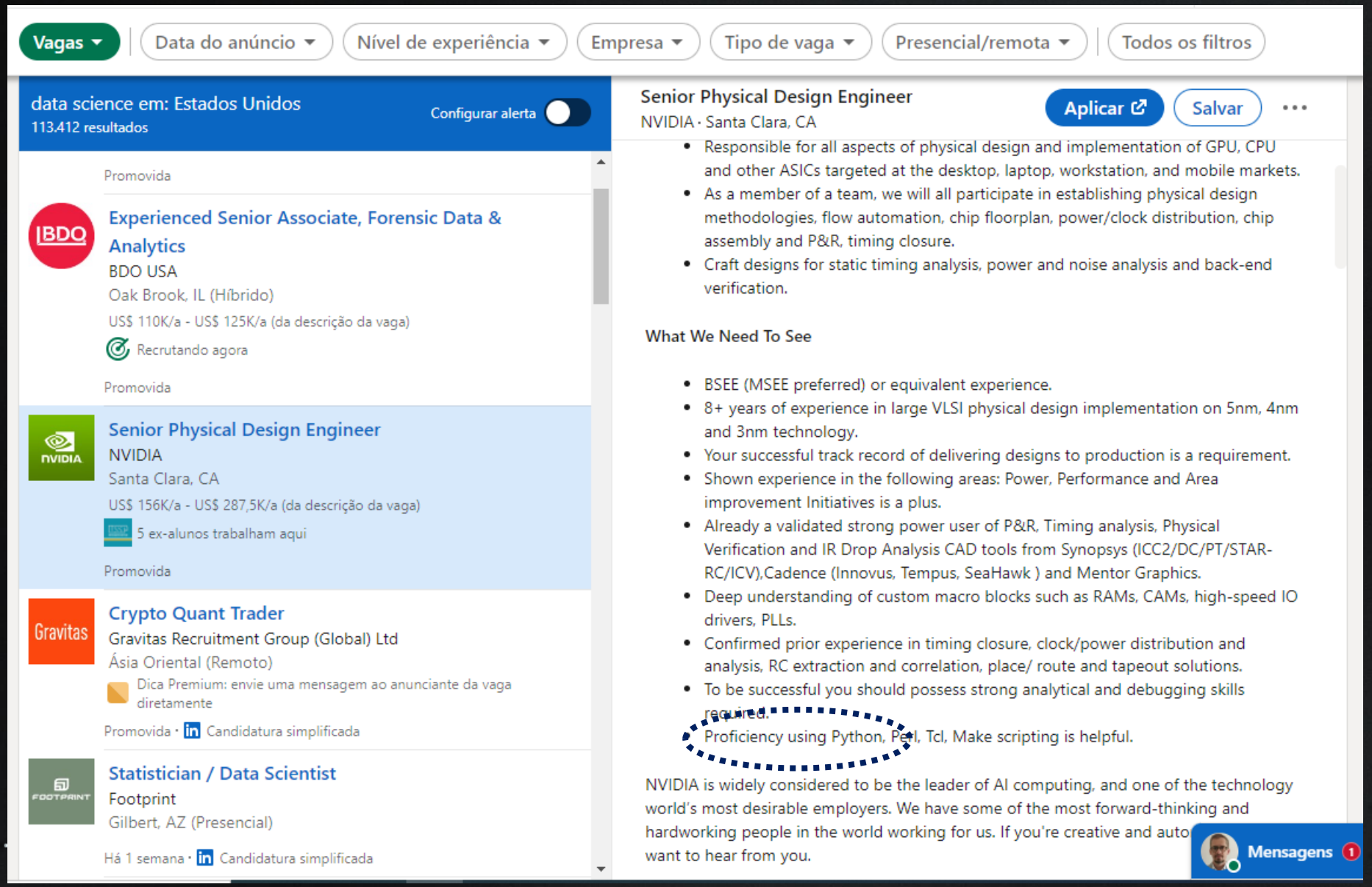


Por que Python está em alta?

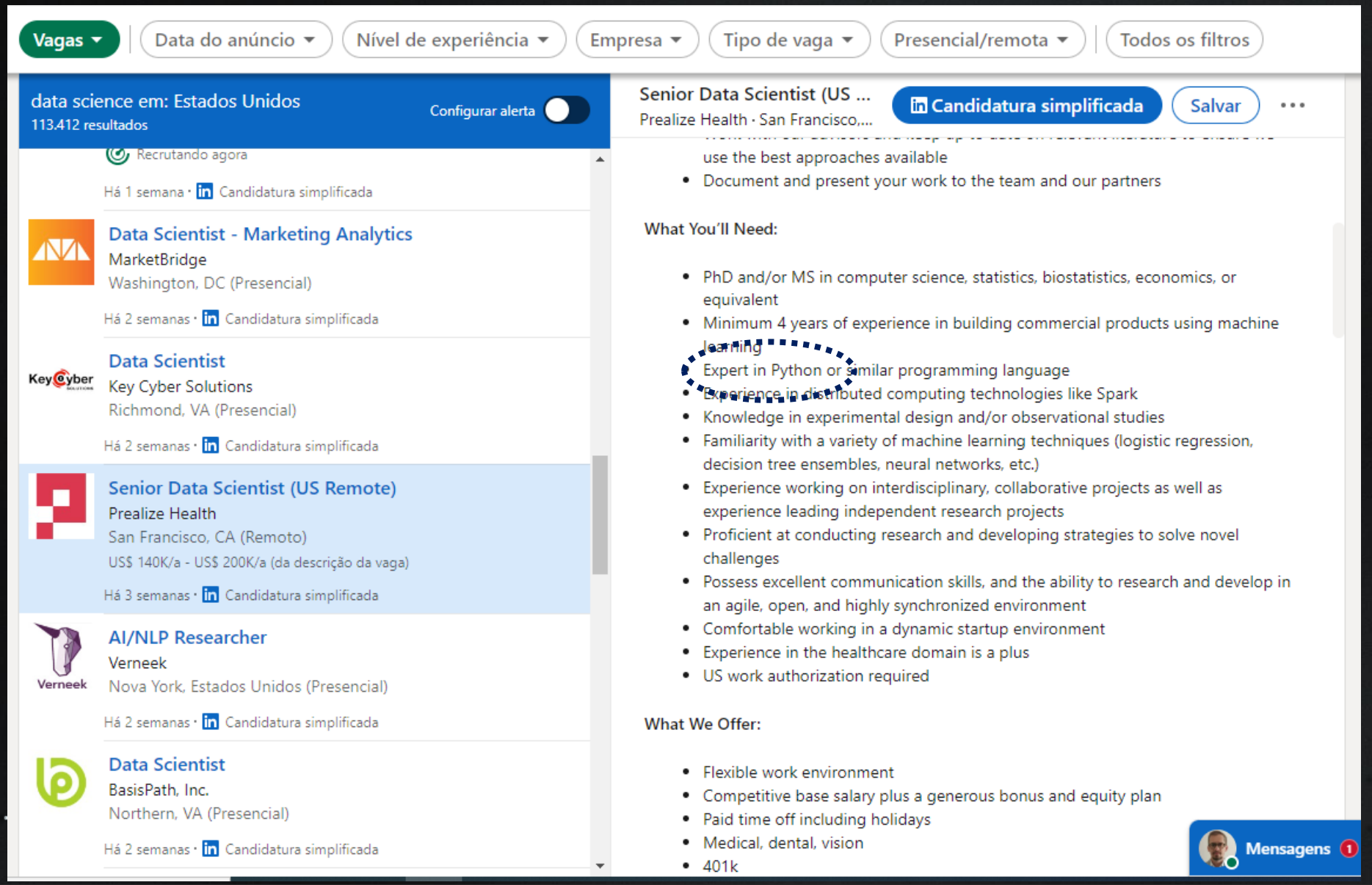
- Sintaxe simples
- Alta segurança
- Diversas Bibliotecas
- Vasta comunidade

- Útil em ML e IA
- Útil em ciência de dados
- Pesquisa científica acadêmica
- Documentação

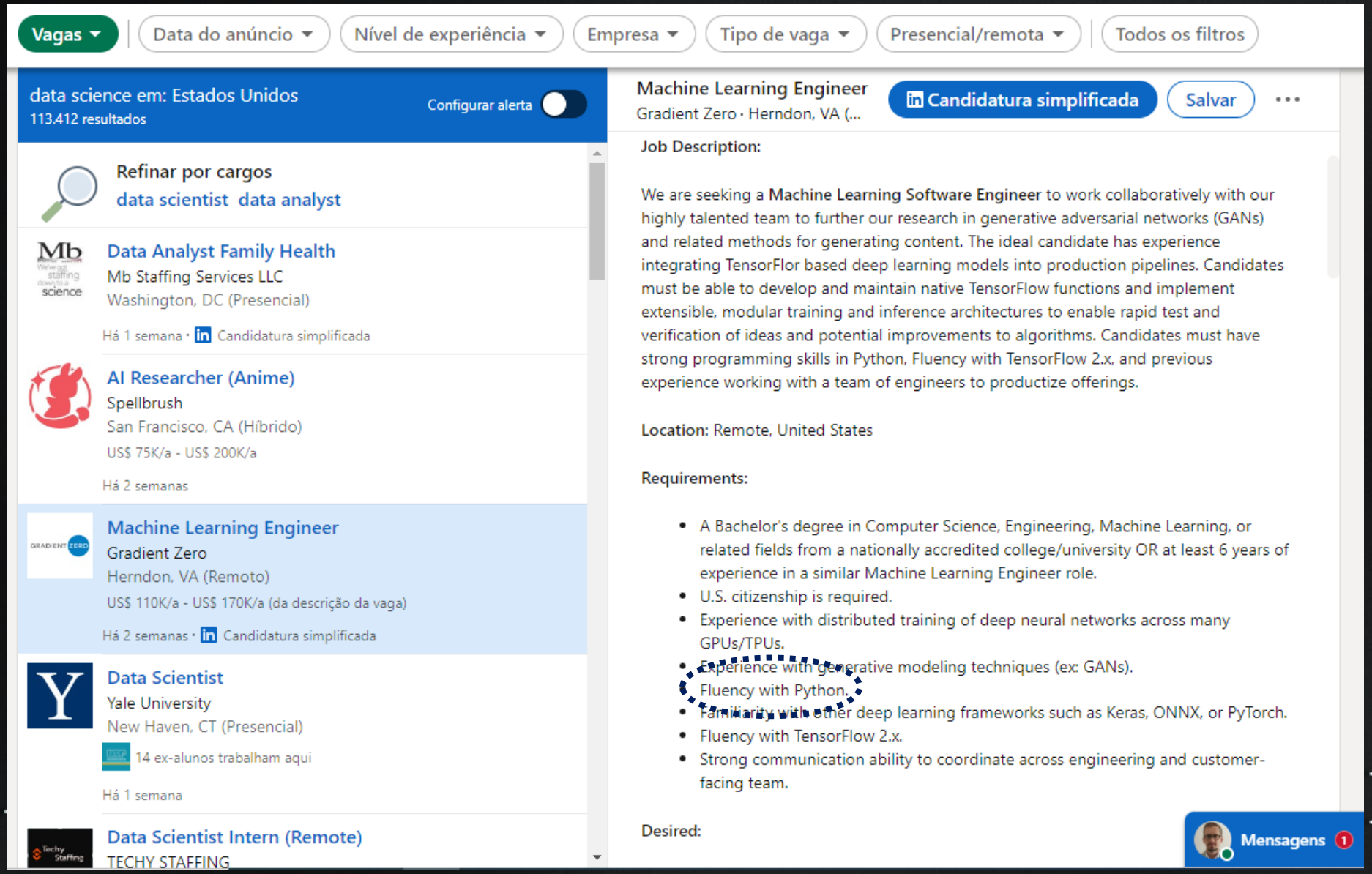
Mercado para desenvolvedores (Python)



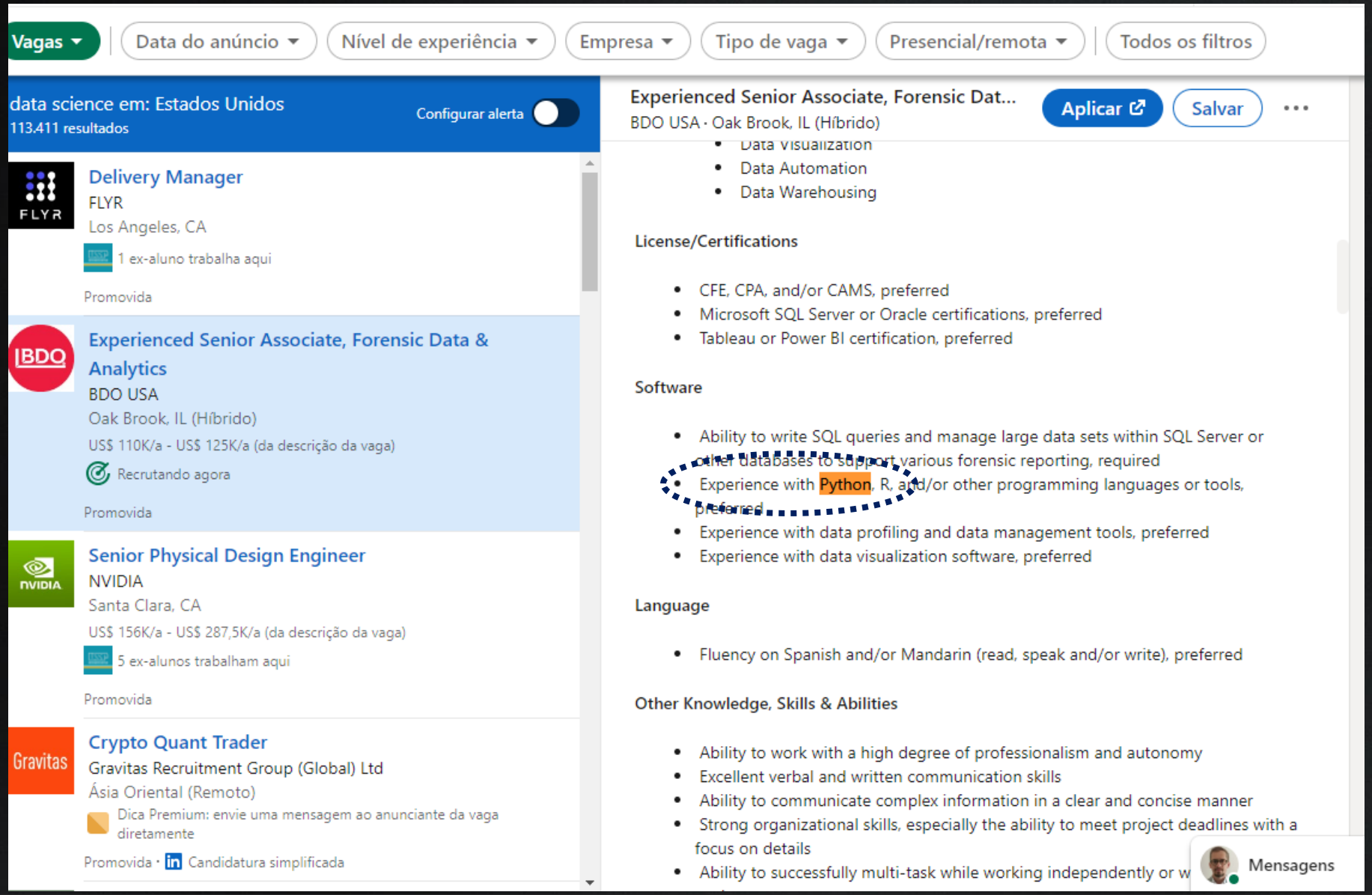
Mercado para desenvolvedores (Python)



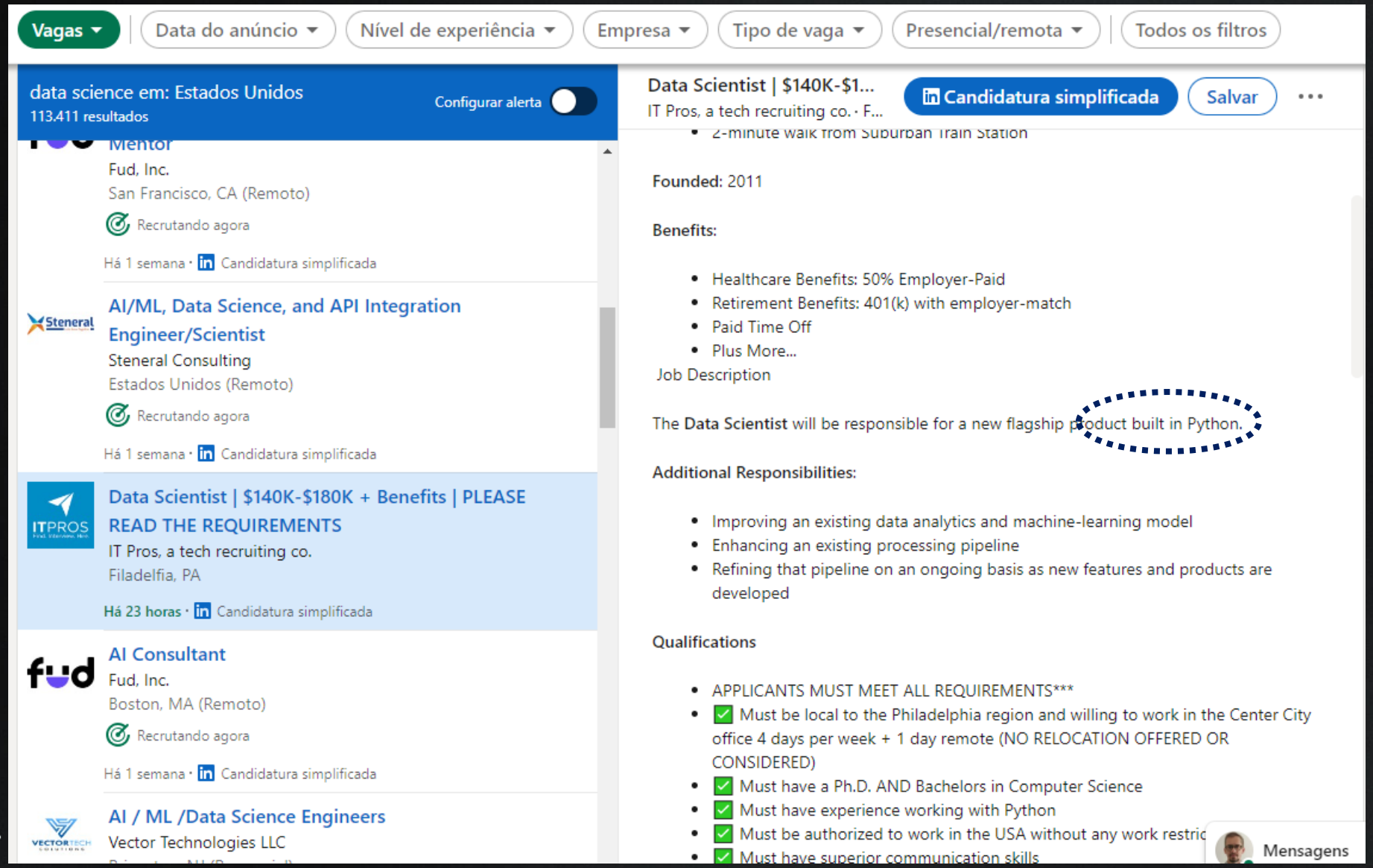
Mercado para desenvolvedores (Python)



Mercado para desenvolvedores (Python)



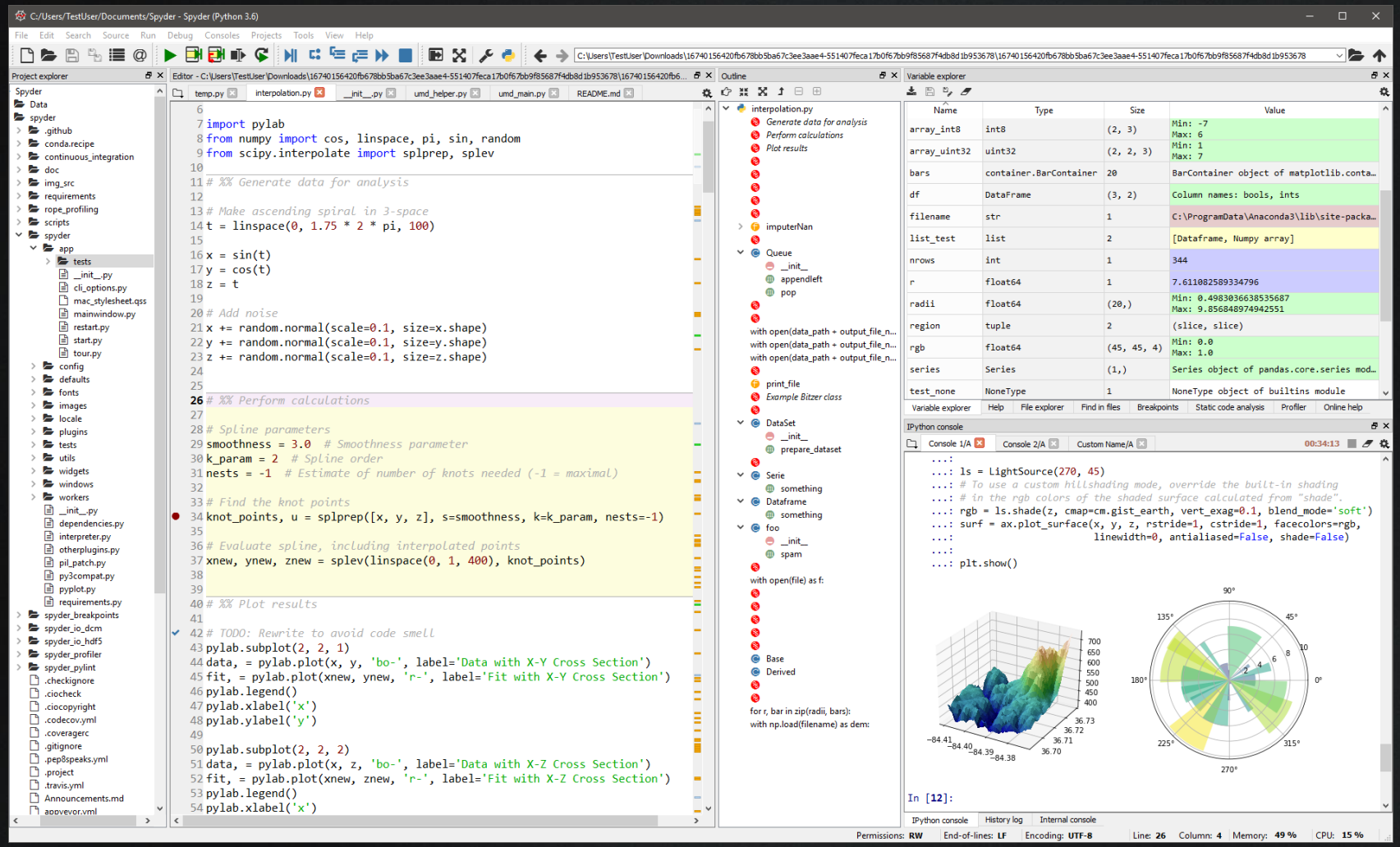
Mercado para desenvolvedores (Python)



Compiladores / Editores de código em Python

Compiladores

SPYDER
The Scientific Python Development Environment



Compiladores

jupyter

In [1]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import plotly
from IPython.display import display, Markdown as md
```

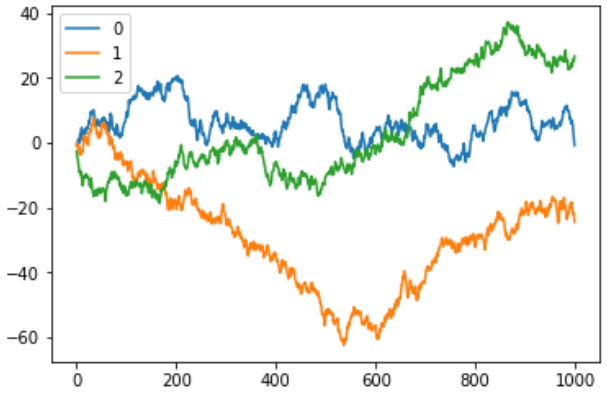
In [2]:
```
title = "My Shiny Report"
x = 1000
y = 3
```

In [3]:
```
display(md("# Just look at this graph from {}".format(title)))
```

Just look at this graph from My Shiny Report

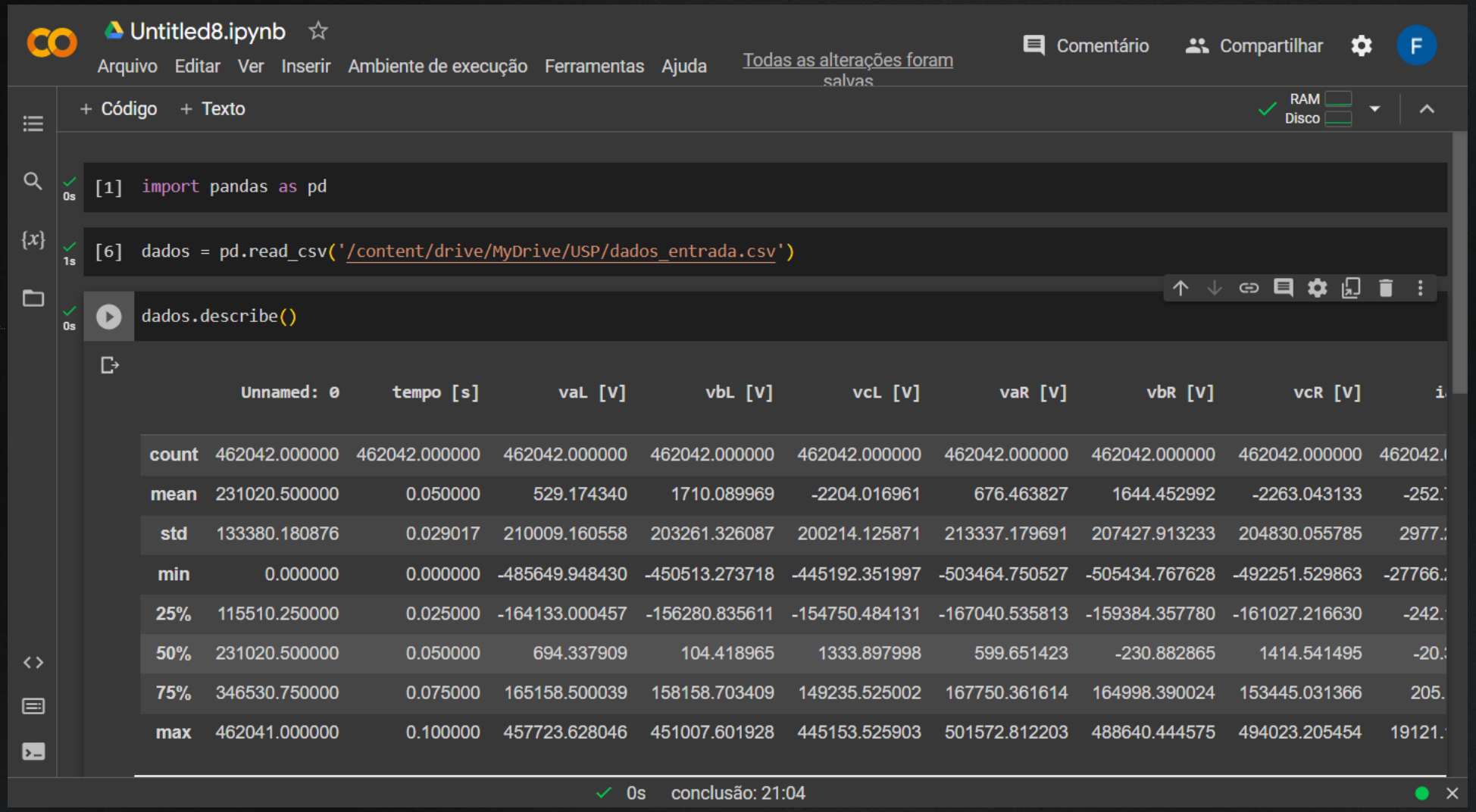
In [4]:
```
df = pd.DataFrame(np.random.randn(x, y))
df.cumsum().plot()
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f127adda278>
```



Compiladores

Google colab

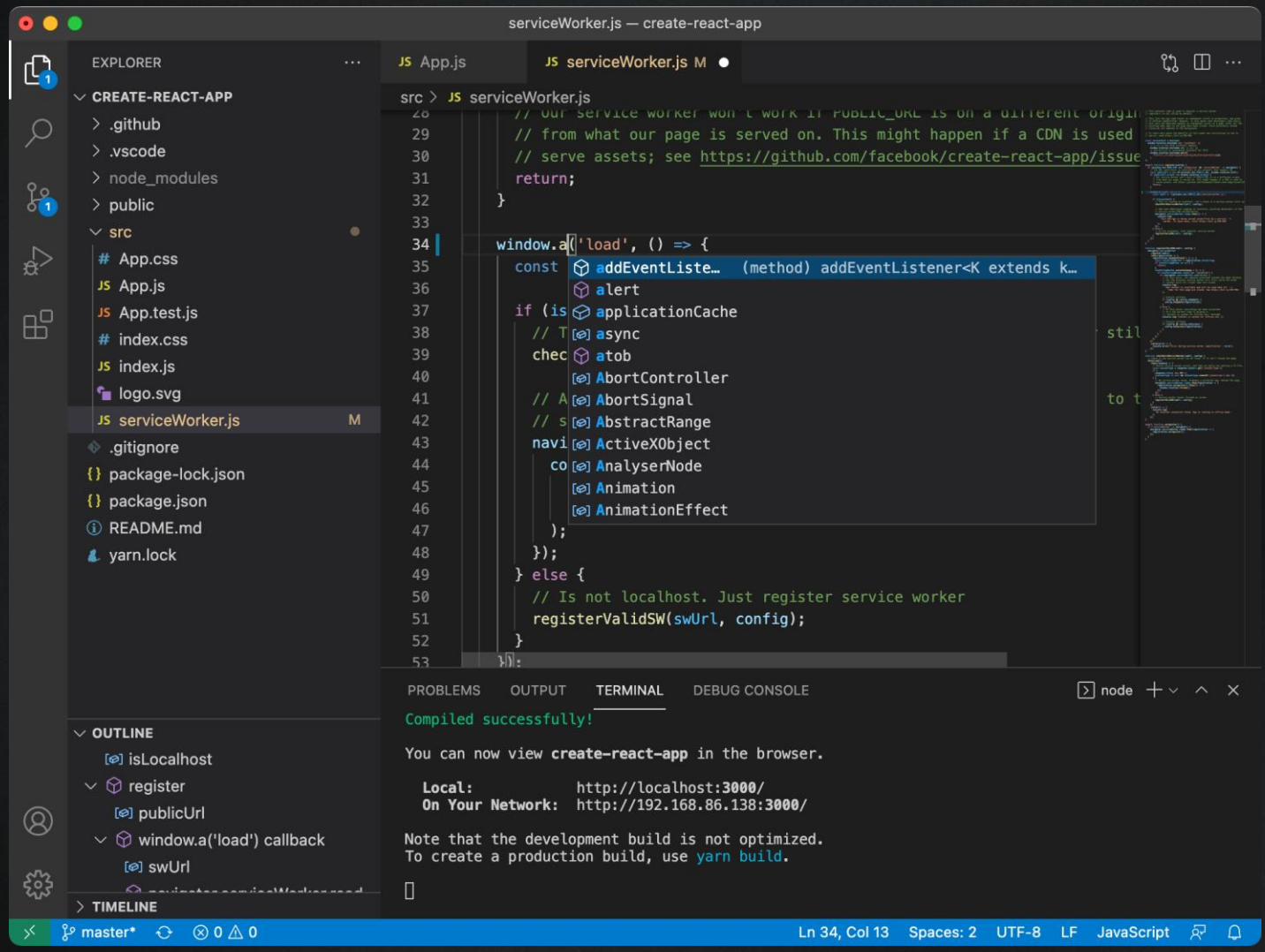


	Unnamed: 0	tempo [s]	vaL [V]	vbL [V]	vcL [V]	vaR [V]	vbR [V]	vcR [V]	i
count	462042.000000	462042.000000	462042.000000	462042.000000	462042.000000	462042.000000	462042.000000	462042.000000	462042.
mean	231020.500000	0.050000	529.174340	1710.089969	-2204.016961	676.463827	1644.452992	-2263.043133	-252.
std	133380.180876	0.029017	210009.160558	203261.326087	200214.125871	213337.179691	207427.913233	204830.055785	2977.
min	0.000000	0.000000	-485649.948430	-450513.273718	-445192.351997	-503464.750527	-505434.767628	-492251.529863	-27766.
25%	115510.250000	0.025000	-164133.000457	-156280.835611	-154750.484131	-167040.535813	-159384.357780	-161027.216630	-242.
50%	231020.500000	0.050000	694.337909	104.418965	1333.897998	599.651423	-230.882865	1414.541495	-20.
75%	346530.750000	0.075000	165158.500039	158158.703409	149235.525002	167750.361614	164998.390024	153445.031366	205.
max	462041.000000	0.100000	457723.628046	451007.601928	445153.525903	501572.812203	488640.444575	494023.205454	19121.

Compiladores

Visual Studio

Este compilador será utilizado nas aulas



Conceitos sobre algoritmos, fluxogramas e pseudocódigo

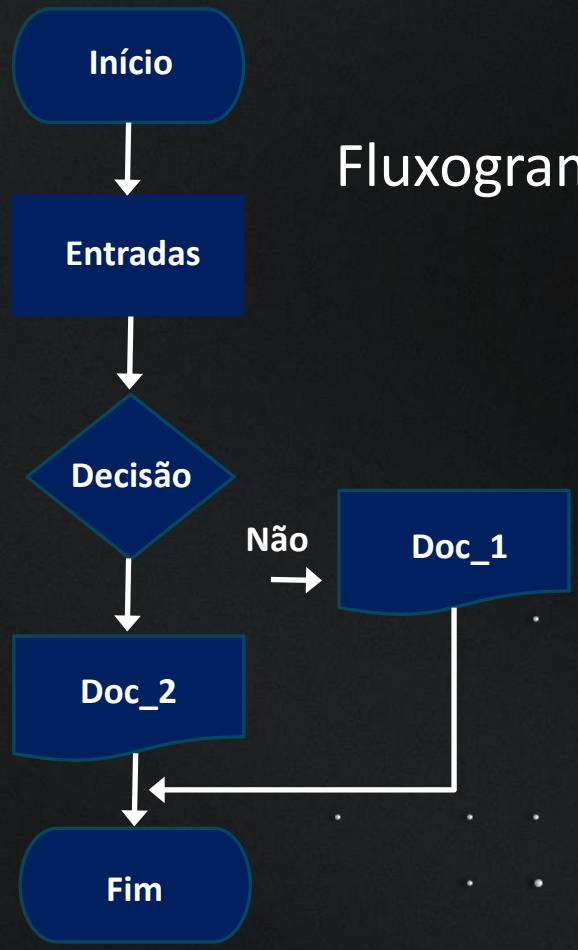
Conceito de Algoritmos

Algoritmos podem ser apresentados por fluxogramas e pseudocódigos.

Pseudocódigo

1. *Ler a primeira nota do aluno (N1)*
2. *Ler a segunda nota do aluno (N2)*
3. *Ler a terceira nota do aluno (N3)*
4. *Calcular a média das notas: media = (N1 + N2 + N3) / 3*
5. *Se media > 7 então*
    6. *Escrever "Aluno aprovado"*
   Senão
    7. *Escrever "Aluno reprovado"*
8. *Fim*

Fluxograma



Conceito Pseudocódigo

É uma forma genérica de escrever um algoritmo (linguagem simples) sem necessidade de conhecer qualquer linguagem de programação.

Média aluno

```
Início
    Ler a primeira nota do aluno (N1)
    Ler a segunda nota do aluno (N2)
    Ler a terceira nota do aluno (N3)
    Calcular a média das notas: media =
(N1 + N2 + N3) / 3
    Se media > 7 então
        Escreva "Aluno aprovado"
    Senão
        Escreva "Aluno reprovado"
Fim
```

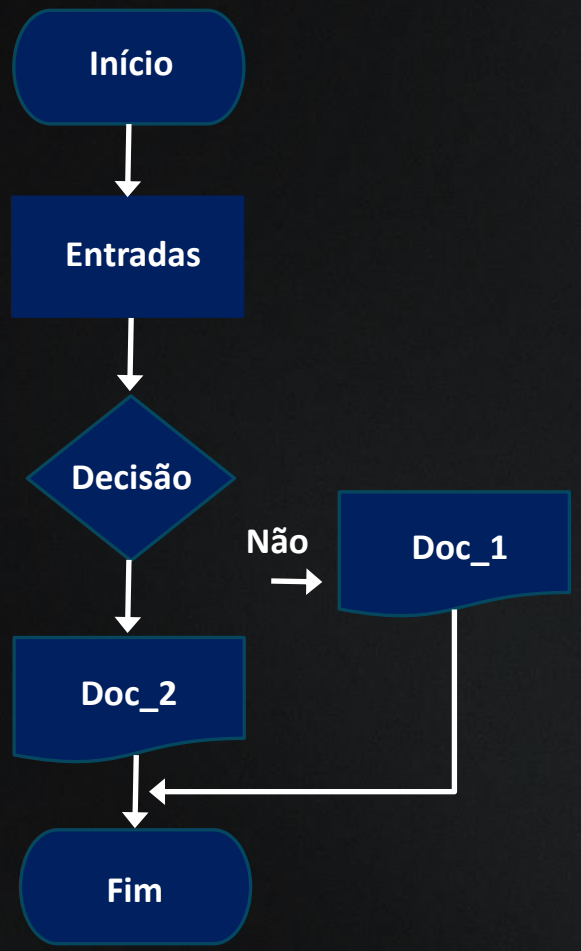
Maior valor inteiro

```
início
    Digite  um número inteiro para A
    Leia (A)
    Digite  um número inteiro para B
    Leia (B)
    Se A > B
        Escreva (“A é maior que B”)
    Senão
        Escreva (“B é maior que A”)
Fim
```

Área Círculo

```
Início
real: AREA, RAIO
escreva (“Digite o raio do círculo em
centímetros: ”)
leia (RAIO)
        AREA ← 3.1416 * (RAIO * RAIO)
{π = 3.1416, aproximado}
escreva (“Área = ”, AREA)
se AREA < 5
 então escreva (“Área pequena”)
Fim
```

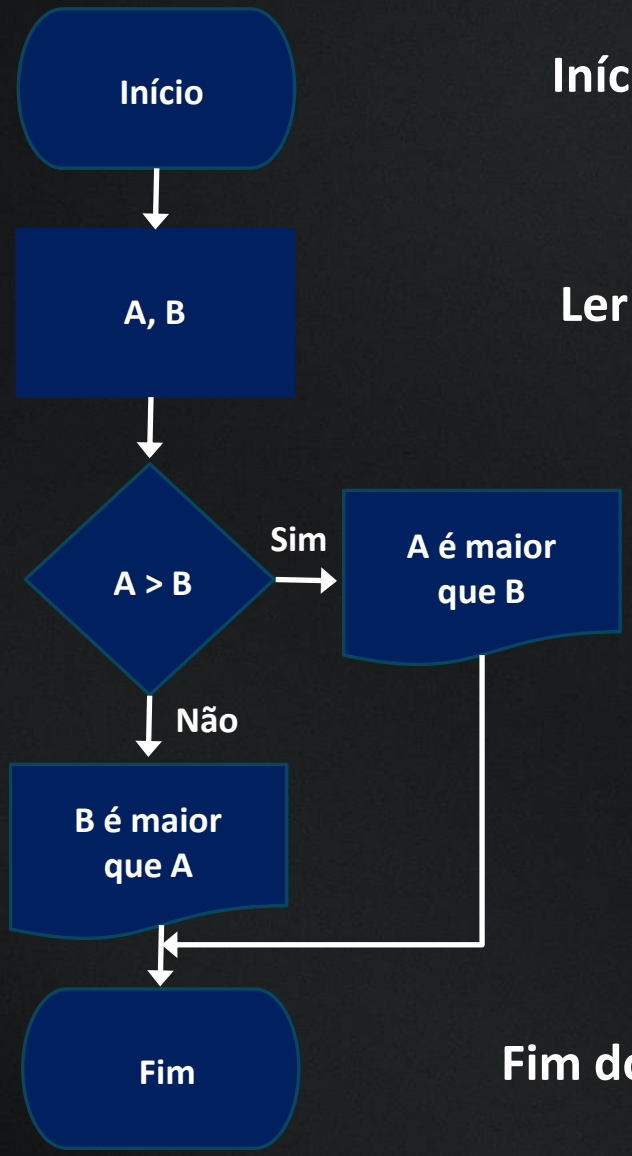
Conceito Fluxograma



É um diagrama que descreve um processo, sistema ou algoritmo de computador. Utiliza alguns símbolos para representar início, entrada e saída, decisão, etc.

São utilizados para documentar, estudar, planejar, melhorar e comunicar processos complexos por meio de diagramas.

Fluxograma – Comparação entre A e B



Início do fluxograma

Ler valores de A e B

Processo de tomada de decisão: verificação da condição entre A e B

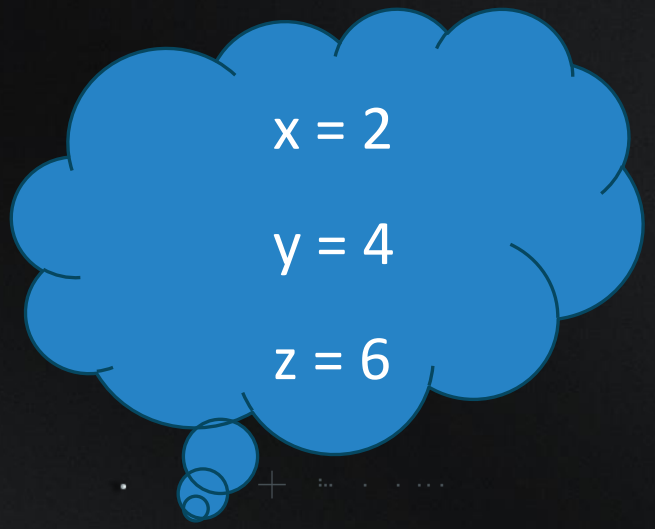
Retorna valor A, pois é maior que B

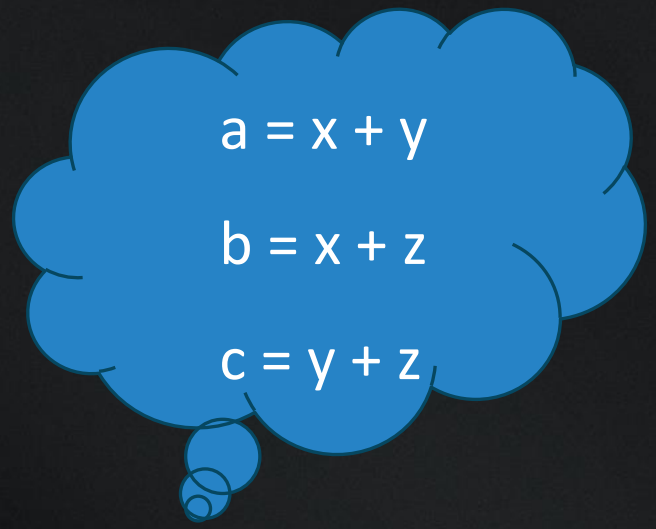
Retorna valor B, pois é maior que A

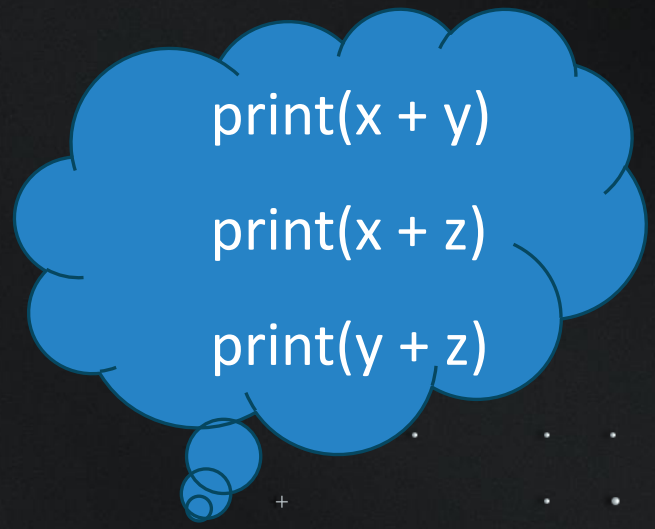
Fim do algoritmo

Conceitos de variáveis e atribuições

O comando de atribuição altera o valor armazenado em uma variável. O valor atribuído torna-se o novo valor da variável, substituindo o anterior. Quando lermos nosso programa, as operações de atribuição serão chamadas de "recebe'; ou seja, uma variável recebe um valor.





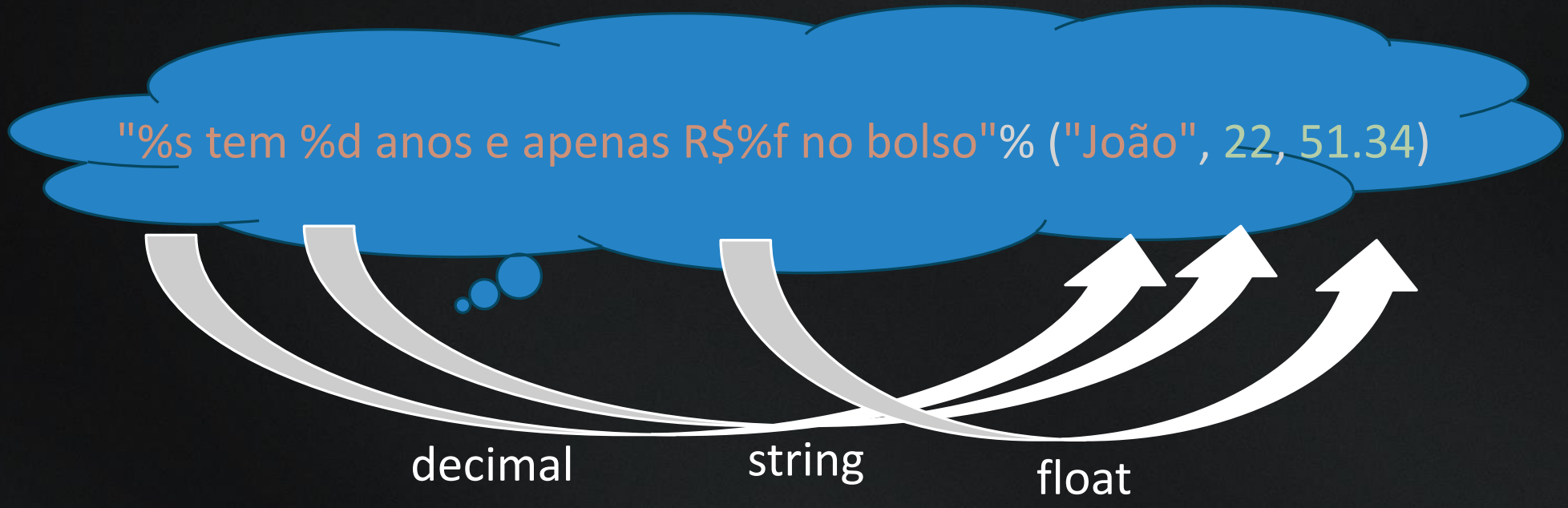


Nome das Variáveis

Nome	Válido	Comentários
A1	Sim	Embora contenha um número, o nome a1 inicia com letra.
Velocidade	Sim	Nome formado por letras.
Velocidade90	Sim	Nome formado por letras e números, mas iniciado por letra.
Salário_médio	Sim	O símbolo sublinha U é permitido e facilita a leitura de nomes grandes.
Salário médio	Não	Nomes de variáveis não podem conter espaços em branco.
_b	Sim	O sublinha U é aceito em nomes de variáveis, mesmo no início.
1a	Não	Nomes de variáveis não podem começar com números.

Operações com strings
Composição

João tem 22 anos e apenas R$51.340000 no bolso



Código pensando em Python

`nota1 = float(input("Digite a primeira nota: "))`	→	Variável nota1
`nota2 = float(input("Digite a segunda nota: "))`	→	Variável nota2
`nota3 = float(input("Digite a terceira nota: "))`	→	Variável nota3
`media = (nota1 + nota2 + nota3) / 3`	→	Variável média
`print("A média das notas é:", media)`	→	Imprimir o valor da variável média

Código pensando em Python

```
A = float(input("Digite o valor de A: "))
B = float(input("Digite o valor de B: "))

if A > B:
    print("A é maior do que B")
else:
    print("B é maior do que A")
```

Variável A

Variável B

Condição Se

Retorna A caso A seja maior que B

Condição caso a condição do if seja falso

Retorna B caso A seja menor que B

Código pensando em Python

```
raio = float(input("Digite o raio do círculo: "))
area = 3.1415 * raio ** 2

print("A área do círculo é:", area)
```

raio = float(input("Digite o raio do círculo: ")) → Variável raio

area = 3.1415 * raio ** 2 → Variável área

print("A área do círculo é:", area) → Retorna área do círculo

Resolução dos Exercícios
Ex. 01

```
m = float(input("Inserir valor da massa em kg: "))
```
→ Condição para faixa adolescente

```
energia = (m * 3 * pow(10,8)) / 1000000
```
→ Calculo da energia

```
print("A energia resultante é de", energia, "Mjoules")
```
→ Saída da variável energia

Resolução dos Exercícios
Ex. 02

Agora crie um algoritmo que imprima a mensagem criança, adolescente, adulto-jovem, meia-idade ou idoso de acordo com as seguintes faixas de idade:

Criança - nascimento até os 11 anos de idade

Adolescência - entre 12 e 20 anos

Adulto-jovem - entre 21 e 40 anos

Meia-idade - entre 40 e 65 anos

Idoso - acima dos 65 anos de idade

Resolução dos Exercícios
Ex. 02

```
idade = int(input("Digite a idade: "))
if idade <= 11:
    print("crianca")
elif idade >= 12 and idade <= 20:
    print("adolescente")
elif idade >= 21 and idade <= 40:
    print("adulto_jovem")
elif idade >= 41 and idade <= 65:
    print("meia_idade")
else:
    print("Idoso")
```

Digitar idade

Condição para faixa criança

Condição para faixa adolescente

Condição para jovem adulto

Condição para meia idade

Condição para faixa idoso

Condições – if/elif/else

Caso a condição if e elif não sejam atendidas, usamos else.

if <condição>:

condição principal

elif <condição>:

condição secundário

else:

caso primeira e segunda

condição não sejam atendidads

```
a = int(input( "Primeiro valor: "))
b = int(input( "Segundo valor: "))
c = int(input( “Terceiro valor: "))
if a > b:
    print( “A é maior que B.")
elif c > a:
    print(“C é maior que A.")
else:
    print(“B é maior que A ou B é maior que C.")
```

Cálculo da área em função do tipo de área a ser calculada

```
import math
pi = math.pi
area_desejada = ['retangulo', 'quadrado', 'circulo']
escolha = (input("Escolha uma forma (retangulo, quadrado, circulo): ")).lower()

if escolha == 'retangulo':
        lado_1 = float(input("Insira o primeiro lado: "))
        lado_2 = float(input("Insira o segundo lado: "))
        print(f"A área deste retângulo é: {lado_1 * lado_2}", "m2")
elif escolha == 'quadrado':
        lado_1 = float(input("Insira o primeiro lado: "))
        lado_2 = float(input("Insira o segundo lado: "))
        print(f"A área deste quadrado é {lado_1 * lado_2}", "m2")
elif escolha == 'circulo':
        raio = float(input("Insira valor do raio: "))
        area = math.pi * (raio ** 2)
        print(f"A área deste círculo é {area:.2f} m2")
else:
    print("Escolha inválida.")
```

Observação

Foi necessário inserir quatro condições utilizando if, dois elif e else.

Operador in

Verifica se a marca de um carro está na lista

```
# Cria uma lista de carros
carros = ['audi', 'bmw', 'subaru', 'toyota’]
# Usuário digita uma marca de carro de sua escolha
carro_digitado = (str(input("Digite o nome da marca do carro")).lower())
# Condição para verificar se carro digitado está na lista
if carro_digitado in carros:
  print("O carro escolhido é:", carro_digitado)
else:
   print("Carro não está na lista!")
```

in
Verifica se um valor está presente em uma sequência

Operador not in

Verifica se uma letra não está na palavra

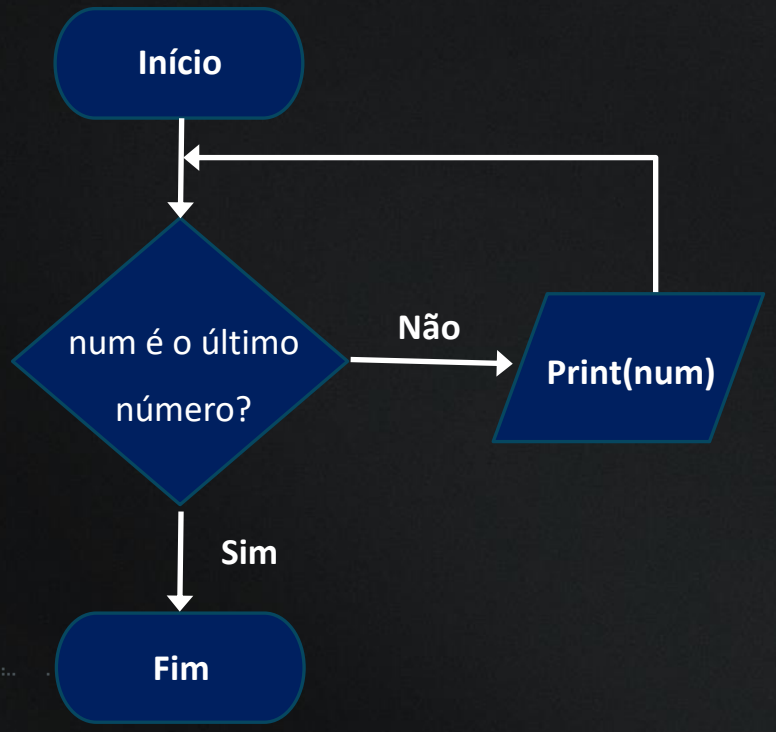
```
# Solicite ao usuário uma letra e uma palavra
letra = input("Digite uma letra: ").lower()
palavra = input("Digite uma palavra: ").lower()

# Verifique se a letra não está na palavra
if letra not in palavra:
    print(f"A letra '{letra}' não está na palavra.")
else:
    print(f"A letra '{letra}' está na palavra.")
```


not in
Verifica se um valor está presente em uma sequência

Instrução for

É uma estrutura de controle de fluxo que permite percorrer elementos em uma sequência (como uma lista, string, etc.) e realizar operações em cada um deles.

"num" se torna o primeiro número em números



O print(numero) é executado enquanto ainda houver números na lista

```
numeros = [1, 2, 3, 4, 5]

for numero in numeros:
    print(numero)
```

Instrução for

É uma estrutura de controle de fluxo que permite percorrer elementos em uma sequência (como uma lista, string, etc.) e realizar operações em cada um deles.

```
coberturas_requisitadas = ['pepperoni', 'tomate', 'queijo_extra', 'cebola']

for cobertura_requisitada in coberturas_requisitadas:
    print(f"Adicione {cobertura_requisitada}.")
print("\nSua pizza esta pronta!")
```

for

Percorre a lista coberturas_requisitadas e realiza uma ação para cada elemento na lista.

Instrução for e continue

```
frutas = ["maça", "banana", "melância"]
for x in frutas:
  if x == "maça":
    continue
  print(x)
```

continue

Com a instrução continue podemos parar a iteração atual do loop e continuar com a próxima

```
frutas = ["maça", "banana", "melância"]

for x in frutas:
  print(x)
  if x == "banana":
    break
```

break

Com a instrução break podemos parar o loop antes que ele percorra todos os itens

Instrução if dentro de um for

E se a pizzaria ficar sem pepperoni? Uma declaração if dentro do loop for pode lidar com esta situação:

```
coberturas_requisitadas = ['pepperoni', 'tomate', 'queijo_extra', 'cebola']
for cobertura_requisitada in coberturas_requisitadas:
    if cobertura_requisitada == 'pepperoni':
        print("Descule, mas estamos sem pepperoni.")
    else:
        print(f"Adicione {cobertura_requisitada}.")
print("\nSua pizza esta pronta!")
```

for e if

Neste caso, uma condição não pode ser atendida. Então, podemos usar um if dentro do for.

Instrução for dentro de um for

Será feito uma combinação de x (elemento da lista ordem) com y (elemento da lista titulo).

```
ordem = ["first", "second", "third", "fourth"]
titulo = ["king", "queen", "princess", "Duke"]

for x in ordem:
    for y in titulo:
        print(x, y)
```

For dentro for

A cada iteração dos loops aninhados, a função print(x, y) é chamada.

Instrução if dentro de um for

```
# Exemplo numérico com um loop for e estrutura if
numeros = [1, 3, 5, 7, 9, 2, 4, 6, 8, 10]

# Loop for para percorrer a lista de números
for numero in numeros:
    if numero % 2 == 0:
        print(f"{numero} é um número par.")
    else:
        print(f"{numero} é um número ímpar.")
```

Número Par ou Ímpar

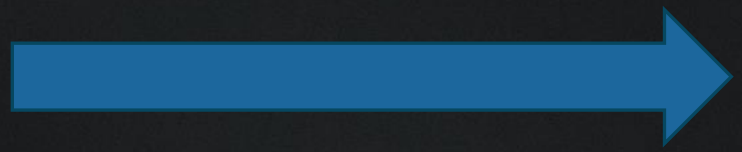
"if“ é utilizado para verificar se o número atual (numero) é divisível por 2 (se é par).

Estrutura de repetição while

Repetições representam a base de vários programas. São utilizadas para executar a mesma parte de um programa várias vezes, normalmente dependendo de uma condição. Por exemplo, para imprimir três números na tela, poderíamos escrever um programa como:

```
x = 1
print(x)
x = x + 1
print(x)
x = x + 1
print(x)
```

Estrutura usando while



```
x = 1

while x <= 3:
    x = x + 1
    print(x)
```

Contadores

Imagine um problema em que deveríamos imprimir os números inteiros entre 1 e um valor digitado pelo usuário. Vamos escrever um programa de forma que o último número a imprimir seja informado pelo usuário:

```
fim = int(input( "Digite o último número a imprimir: "))
x = 1

while x <= fim:
    x = x + 1
    print(x)
```

Imprime os valores de 1 até o valor digitado pelo usuário

Contadores

Faça um algoritmo para imprimir a tabuada de adição de um número digitado pelo usuário. Essa tabuada deve ser impressa de 1 a 10, sendo no número digitado pelo usuário. Teríamos, n+ 1, n+2, ... n+ 10.

```
n = int(input( "Tabuada de:"))
x = 1
while x <= 5:
    print(n + x)
    x = x + 1
```

Loop dentro de um Loop
Usando break

```
for i in range(2, 4):

    for j in range(1, 11):
     if i==j:
       break
     print(i, "*", j, "=", i*j)
    print()
```

Primeira Iteração	*Segunda Iteração*	*i * j*
2	1	2
	2	i == j
3	1	3
	2	6
	3	i == j

i varia de 2 a 3 e j varia de 1 a 10. Se i for igual a j, o loop interno é interrompido e o programa passa para a próxima iteração do loop externo.

Loop dentro de um Loop
Usando continue

```
for i in range(2, 4):

    for j in range(1, 11):
      if i==j:
        continue
      print(i, "*", j, "=", i*j)
```

i varia de 2 a 3 e j varia de 1 a 10. Se i for igual a j, não há iteração entre i e j. Porém, a iteração continua até para i e j.

Primeira Iteração	*Segunda Iteração*	*i * j*
2	1	2
	2	i == j
	3	6
	⋮	-
	10	20
3	1	3
	2	6
	3	i == j
	⋮	-
	10	30

Loop dentro de um Loop
Exemplo três listas

```
alunos = ['João', 'Maria', 'Pedro', 'Ana']
notas = [8.5, 9.0, 7.5, 9.5]
faltas = [2, 0, 1, 3]

for i in range(len(alunos)):
    if notas[i] >= 8.0 and faltas[i] == 0:
```

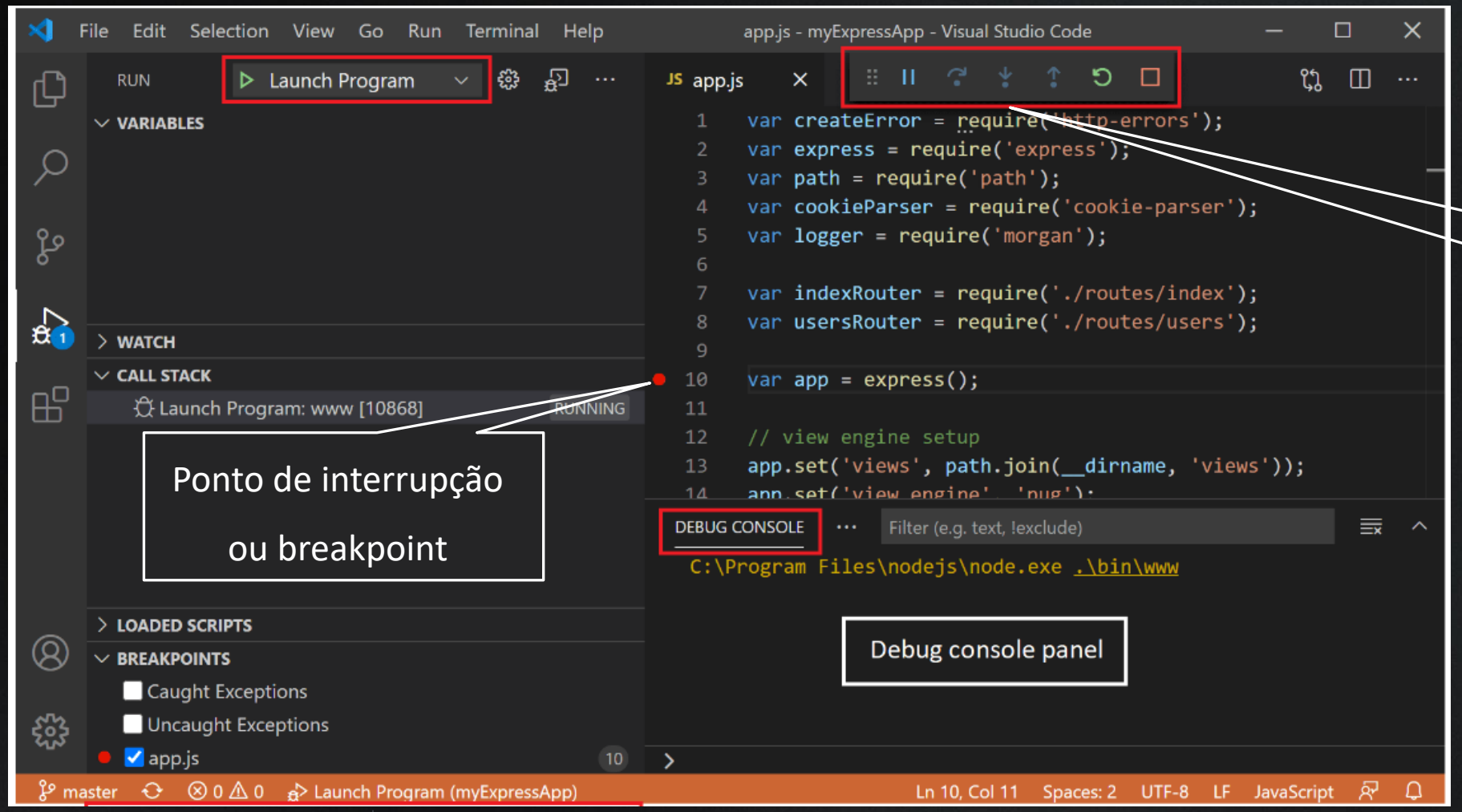
Itera sobre os índices dos elementos nas listas alunos, notas e faltas.

len(alunos) retorna o número de elementos na lista alunos, que é o número total de alunos.

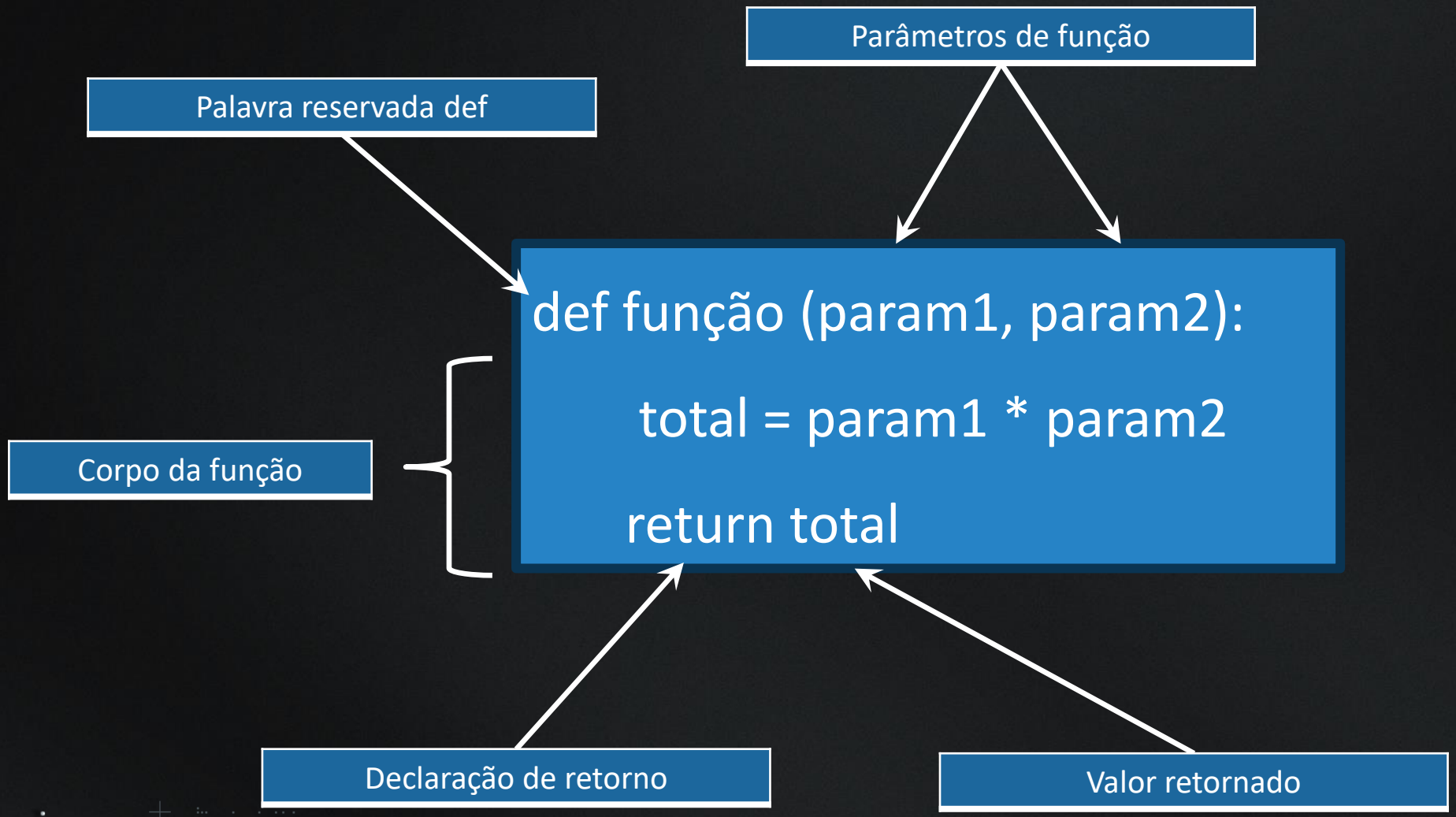
notas[i] >= 8.0: Verifica se a nota do aluno na posição i é maior ou igual a 8.0.

faltas[i] == 0: Verifica se o número de faltas do aluno na posição i é igual a zero.

Debug ou Depuração no VSCode



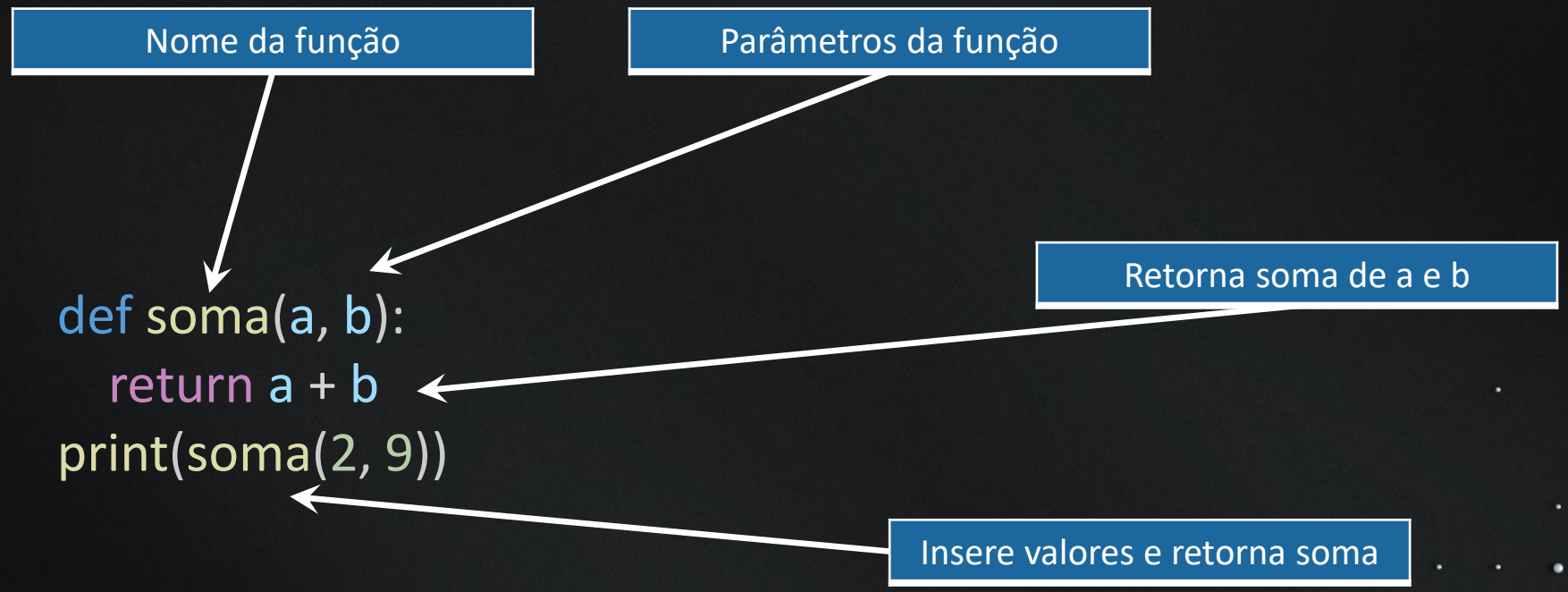
Anatomia das Funções



Retornando uma função em Python

Para retornar uma função em Python é necessário inserir o nome da função.

Para o exemplo abaixo, soma() e dois valores entre o parêntese.



Aplicação de funções em dataframes

As funções definidas pelo usuário (def) em um DataFrame no contexto do pandas em Python oferece diversos benefícios, proporcionando uma abordagem mais modular e legível para a manipulação e análise de dados.

- Legibilidade
- Abstração de Lógica
- Facilidade de Testes
- Manutenção Simplificada
- Encapsulamento de Operações Complexas

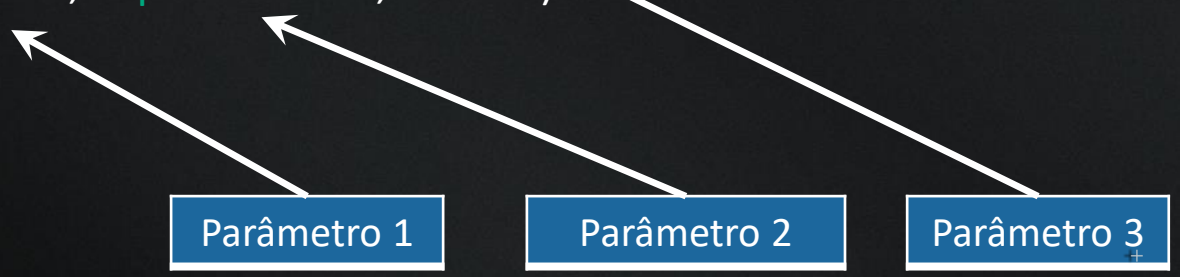
Dataframe

```python
import pandas as pd

# Dados
dados = {
    'Tipo de Carro': ['Sedan', 'SUV', 'Hatchback', 'Caminhonete', 'Coupé'],
    'Tipo de Direção': ['Hidráulica', 'Elétrica', 'Mecânica', 'Elétrica', 'Hidráulica'],
    'Tipo de Pneus': ['Radial', 'Diagonal', 'Radial', 'Diagonal', 'Radial'],
    'Faixa de Preço': ['Alto', 'Médio', 'Baixo', 'Alto', 'Médio'],
    'Potência - CV': [120, 130, 100, 180, 200],
    'Velocidade de Cruzeiro': [True, False, True, True, False]
}

df_carros = pd.DataFrame(dados)
```

Filtragem por valor

Crie uma função chamada filtra_por_valor que recebe um DataFrame, o nome de uma coluna e um valor como parâmetros. A função deve retornar um novo DataFrame contendo apenas as linhas onde a coluna especificada é igual ao valor fornecido.

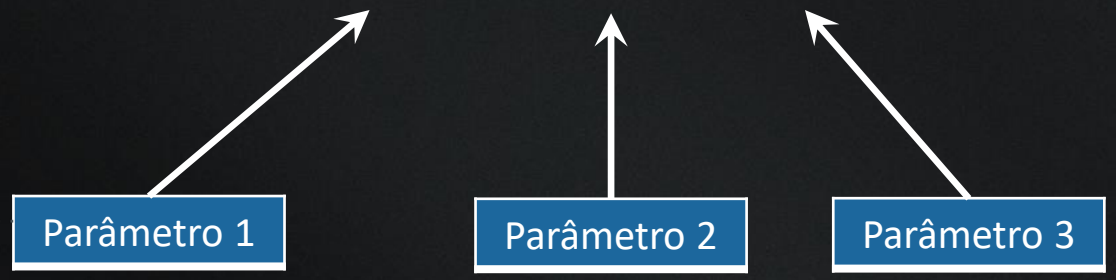
```python
import pandas as pd

def filtra_por_valor(df_carros, coluna, valor):
    return df_carros[df_carros[coluna] == valor]

filtra_por_valor(df_carros, 'Tipo de Carro', 'Sedan')
```



Substituição de Valores

Crie uma função chamada substitui_valores que recebe um DataFrame, o nome de uma coluna, um valor antigo e um valor novo como parâmetros. A função deve substituir todos os valores antigos pelo valor novo na coluna especificada.

```
import pandas as pd

def substitui_valores(df, coluna, valor_antigo, valor_novo):
    df[coluna].replace(valor_antigo, valor_novo)

substitui_valores(df_carros, 'Tipo de Pneus', 'Diagonal', 'Radial')
```



Ordenação por Coluna

Crie uma função chamada ordena_por_coluna que recebe um DataFrame e o nome de uma coluna como parâmetros. A função deve retornar o DataFrame ordenado em ordem crescente com base na coluna especificada.

```
import pandas as pd

def ordena_por_coluna(df, coluna):
    return df.sort_values(by=coluna)

ordenado = ordena_por_coluna(df_carros, 'Potência - CV')
ordenado
```

Para ordenar dados de forma decrescente

ascending=False

Parâmetro 1

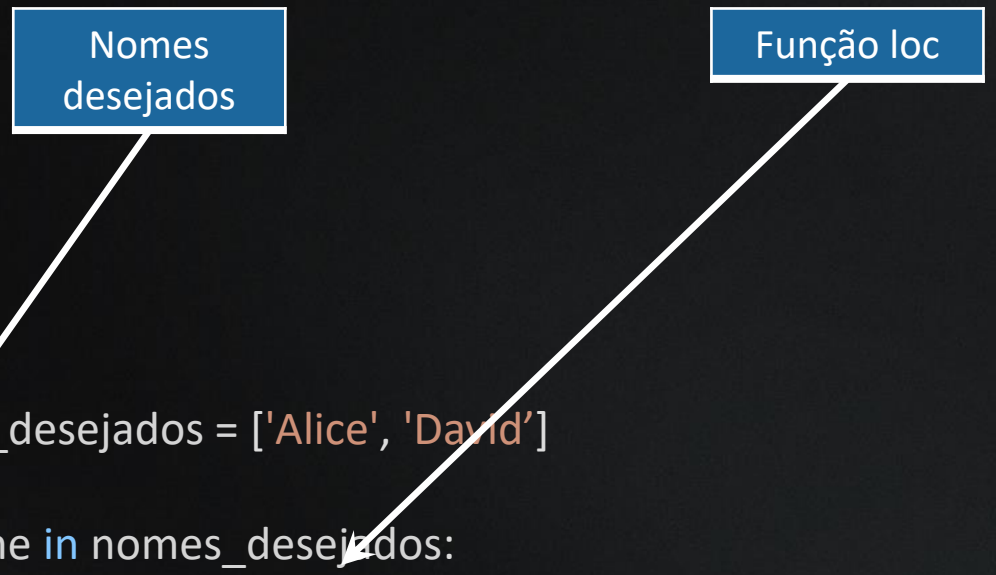
Parâmetro 2

Usando condições em um dataframe

```
import pandas as pd

data = {
   'Nome': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
   'Idade': [25, 30, 28, 22, 35],
   'Altura': [1.65, 1.78, 1.70, 1.75, 1.68],
   'Cidade': ['São Paulo', 'Rio de Janeiro', 'Belo Horizonte', 'Porto Alegre', 'Salvador'],
   'Salario': [50000, 60000, 55000, 48000, 70000]
}

df = pd.DataFrame(data)
df
```

Usando for em um dataframe

Mostre o uso do for para nome alice or david, retorne as idades, altura e cidade.




```
nomes_desejados = ['Alice', 'David']

for nome in nomes_desejados:
    dados = df.loc[df['Nome'] == nome]
    print(f"Dados para {nome}:\n{dados}\n")
```

A função loc é usada para acessar um grupo de linhas e colunas por meio de rótulos ou uma matriz booleana.

Usando while em um dataframe

Percorra o dataframe para idade <=30 e retorne as informações relacionada a cada nome.

```
index = 0

while df['Idade'].iloc[index] <= 30:
    print(df.iloc[index])
    index += 1
```

Usando while em um dataframe

Retorne todas informações do dataframe que possuem cidade a São Paulo.

```
index = 0

while index < len(df):
    if df['Cidade'].iloc[index] == 'São Paulo':
        print(df.iloc[index])
    index += 1
```

Usando while em um dataframe

Retorne as informações de nome e salário que possuem cidade São Paulo.

```
index = 0
while index < len(df):
        if df['Cidade'].iloc[index] == 'São Paulo’:
                print(f"Nome: {df['Nome'].iloc[index]}, Salário: {df['Salario'].iloc[index]}")
        index += 1
```

Usando while em um dataframe

Retorne as informações que possuem cidade de São Paulo em um dataframe.

```
condicao_sao_paulo = df['Cidade'] == 'São Paulo’

df_sao_paulo = df[condicao_sao_paulo]

print(df_sao_paulo)
```

Condição If e Else em um dataframe

Retorne as informações que possuem cidade de São Paulo em um dataframe e crie uma nova variável chamada nascido inserindo Osasco para São Paulo e Guarulhos para outros estados.

```
df['Nascido'] = df['Cidade'].apply(lambda x: 'Osasco' if x == 'São Paulo' else 'Guarulhos')
```

A função apply em pandas é usada para aplicar uma função ao longo de um eixo de um DataFrame. A função pode ser aplicada tanto às linhas quanto às colunas do DataFrame. Ela é especialmente útil quando você deseja realizar operações personalizadas em seus dados.

A palavra-chave lambda em Python é usada para criar funções anônimas, ou seja, funções sem nome.

Resolução de Exercícios

Resolução dos Exercícios
Ex. 03

Uma determinada faculdade utiliza quatro atividades para avaliar o desempenho do seu aluno. No entanto, para verificar o seu desempenho no semestre utiliza apenas as três maiores notas. Crie um algoritmo para calcular a média deste aluno.

Resolução dos Exercícios
Ex. 03

```
Inserir quatro notas
nota1 = float(input("Digite a primeira nota: "))
nota2 = float(input("Digite a segunda nota: "))
nota3 = float(input("Digite a terceira nota: "))
nota4 = float(input("Digite a quarta nota: "))

Calcular a média descartando a menor nota
notas = [nota1, nota2, nota3, nota4]
nota_minima = min(notas)
notas.remove(nota_minima)
media = sum(notas) / 3

print(f"A média final do aluno é: {media:.2f}")
```

Resolução dos Exercícios
Ex. 04

Crie um algoritmo para calcular a potência em watts de um motor em corrente contínua de acordo com as equações abaixo. Repare que o usuário pode inserir tensão V e corrente I, ou Resistência R e Corrente I, ou Tensão V e Resistência R.

P = V x I

P = R x I x I

P = V^2 / R

Resolução dos Exercícios
Ex. 04

```python
corrente = float(input("Inserir valor de corrente: "))
resistencia = float(input("Inserir valor de resistência: "))
tensao = float(input("Inserir valor de tensão: "))

print("valor da corrente", corrente)
print("valor da resistência", resistencia)
print("valor de tensão", tensao)

if tensao == 0 and corrente == 0:
    print("Não é possível calcular a potência, pois tensão e corrente são iguais a zero.")
elif corrente == 0 and resistencia == 0:
    print("Não é possível calcular a potência, pois corrente e resistência são iguais a zero.")
elif resistencia == 0 and tensao == 0:
    print("Não é possível calcular a potência, pois resistência e tensão são iguais a zero.")
elif tensao == 0:
    print((corrente ** 2) * resistencia)
elif corrente == 0:
    print((tensao ** 2) / resistencia)
elif resistencia == 0:
    print(tensao * corrente)
else:
    print("Valores de corrente, resistência e tensão não podem ser todos zero.")
```

“A boa sorte muitas vezes acontece quando a oportunidade se encontra com o preparo.”
(Thomas Edison)

Referências

- ASCENCIO, A. F. G, CAMPOS, E. A. V. Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java, 2ª Edição, São Paulo: Pearson 2007.
- FURGERI, Sérgio. Introdução à Programação em Python. São Paulo: Editora Senac, 2021.
- MENEZES, Nilo. Introdução à Programação em Python. São Paulo: Novatec, 2019
- SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madson. Algoritmos. São Paulo: Pearson, 2004.