

RODRIGO VIANNINI

POSTECH

MACHINE LEARNING ENGINEERING

PYTHON PARA ML E IA

AULA 02

SUMÁRIO

O QUE VEM POR AÍ?	3
HANDS ON	4
SAIBA MAIS.....	5
MERCADO, CASES E TENDÊNCIAS	26
O QUE VOCÊ VIU NESTA AULA?	31
REFERÊNCIAS.....	32

EMSE

O QUE VEM POR AÍ?

Agora que já conhecemos a disciplina, instalamos o PyCharm e nos familiarizamos com o ambiente do Google Colab, é hora de irmos para a prática!

Nesta aula iremos conhecer os tipos de dados, operadores e estruturas de controle, criar funções e métodos e conhecer algumas funções nativas do Python.

EMEND

HANDS ON

Os elementos essenciais desta aula serão discutidos a seguir. Leiam com atenção, assistam as aulas quantas vezes forem necessárias, pratiquem ativamente, evitem copiar códigos e esclareçam suas dúvidas. Aprender a programar só é possível por meio da prática.



SAIBA MAIS

Tipos de dados

Existem vários tipos de dados que podem ser utilizados para armazenar e manipular informações. Vamos conhecê-los:

Inteiros (int): representam os números inteiros, positivos ou negativos, e não incluem frações.

Exemplo:

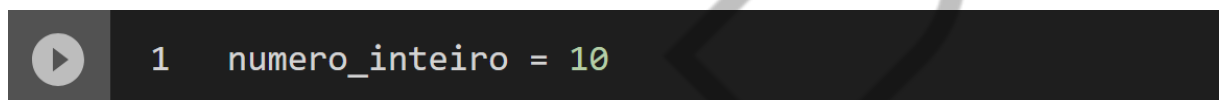
A imagem mostra uma barra de código com fundo escuro. À esquerda, há um ícone de play dentro de um círculo cinza. À direita, o código Python '1 numero_inteiro = 10' está escrito em uma fonte de monoespaço, com o '1' em cinza, 'numero_inteiro' em verde e '10' em amarelo.

Figura 1 – Exemplo de número inteiro
Fonte: Elaborado pelo autor (2024)

Ponto flutuante (float): representam os números reais, incluindo os decimais.

Exemplo:

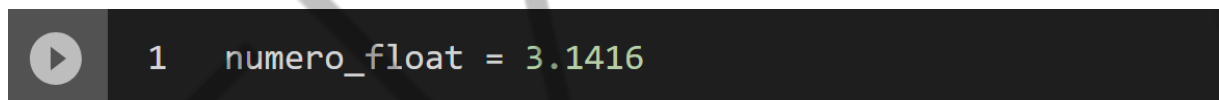
A imagem mostra uma barra de código com fundo escuro. À esquerda, há um ícone de play dentro de um círculo cinza. À direita, o código Python '1 numero_float = 3.1416' está escrito em uma fonte de monoespaço, com o '1' em cinza, 'numero_float' em verde e '3.1416' em amarelo.

Figura 2 – Exemplo de ponto flutuante
Fonte: Elaborado pelo autor (2024)

Caracteres ou strings (str): representam um ou mais caracteres, formando palavras, frases e textos, entre outros.

Exemplo:

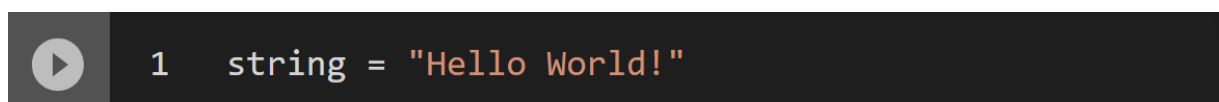
A imagem mostra uma barra de código com fundo escuro. À esquerda, há um ícone de play dentro de um círculo cinza. À direita, o código Python '1 string = "Hello World!"' está escrito em uma fonte de monoespaço, com o '1' em cinza, 'string' em verde e '"Hello World!"' em amarelo.

Figura 3 – Exemplo de string
Fonte: Elaborado pelo autor (2024)

Booleanos (bool): representam os valores verdadeiros (True) ou falsos (False); são utilizados frequentemente em estruturas lógicas.

Exemplo:

```
1 booleano = True
```

Figura 4 – Exemplo de booleano
Fonte: Elaborado pelo autor (2024)

Listas (list): são coleções de dados que podem ser sequenciais ou não. Nelas, é possível encontrar diferentes tipos de dados dentro da mesma lista ou até mesmo uma lista dentro de outra lista (lista aninhada). Podemos declarar uma lista como vazia para depois incluirmos itens de acordo com nossa necessidade. Um fator característico importante é a mutabilidade, ou seja: podemos alterar, adicionar ou remover seus elementos.

Exemplo:

```
1 lista = [1, 2, "três", True]
```

Figura 5 – Exemplo de lista
Fonte: Elaborado pelo autor (2024)

Tuplas (tuple): são semelhantes às listas mas imutáveis, ou seja: não podemos alterar, adicionar ou remover seus elementos.

Exemplo:

```
1 tupla = (1, 2, "três", True)
```

Figura 6 – Exemplo de tupla
Fonte: Elaborado pelo autor (2024)

Conjuntos (set): são coleções não ordenadas de itens únicos. Não considera booleanos.

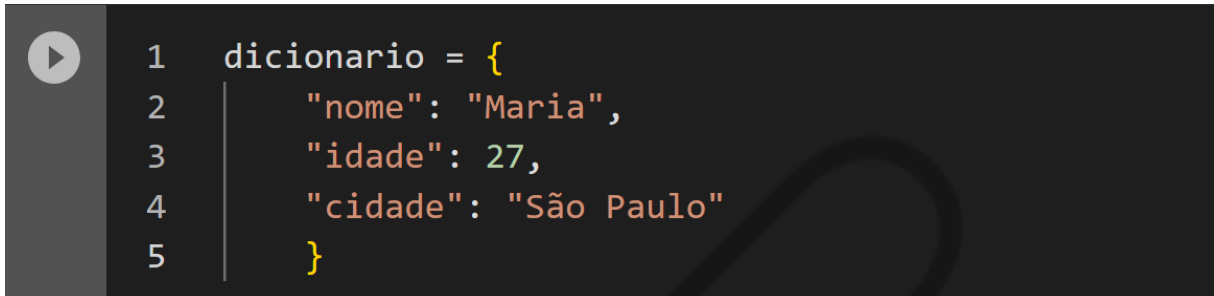
Exemplo:

```
1 conjuntos = {1, 2, "três"}
```

Figura 7 – Exemplo de conjunto
Fonte: Elaborado pelo autor (2024)

Dicionários (dict): armazenam pares de chave-valor, permitindo associar valores às chaves para estruturação interna de dados.

Exemplo:

A imagem mostra um editor de código com um fundo escuro. À esquerda, há uma barra lateral cinza com um ícone de play. O código Python está escrito em uma fonte monoespaçada com coloração sintática: 'dicionario' em cinza, '=' em amarelo, as chaves de dicionário em cor-de-rosa, as strings em verde e os números em verde. O código cria um dicionário com as chaves 'nome', 'idade' e 'cidade' e seus respectivos valores.

```
1 dicionario = {  
2     "nome": "Maria",  
3     "idade": 27,  
4     "cidade": "São Paulo"  
5 }
```

Figura 8 – Exemplo de dicionário
Fonte: Elaborado pelo autor (2024)

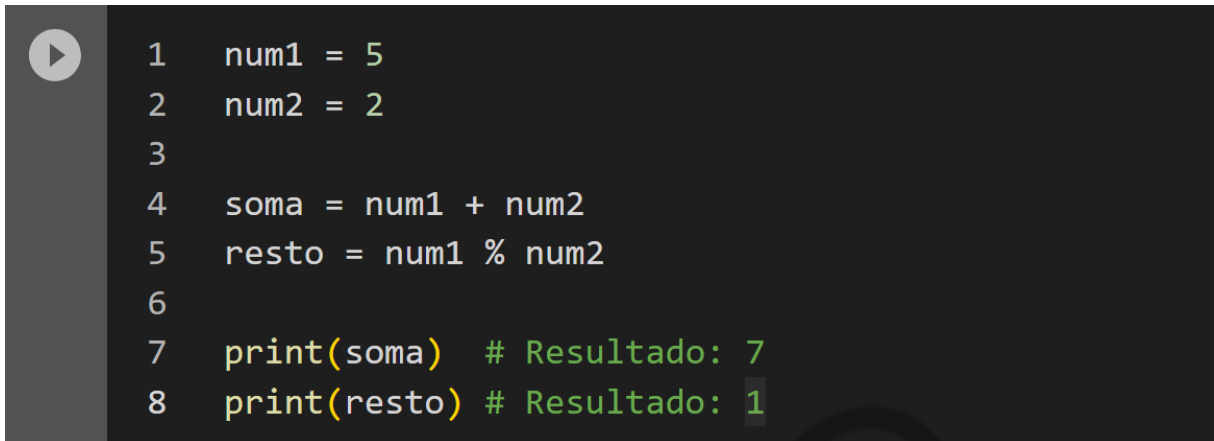
Operadores

Os operadores em Python são símbolos especiais que realizam operações matemáticas/lógicas em variáveis e valores.

Operadores aritméticos:

- (+) adição
- (-) subtração
- (*) multiplicação
- (/) divisão
- (%) resto da divisão
- (//) divisão inteira

Exemplo:



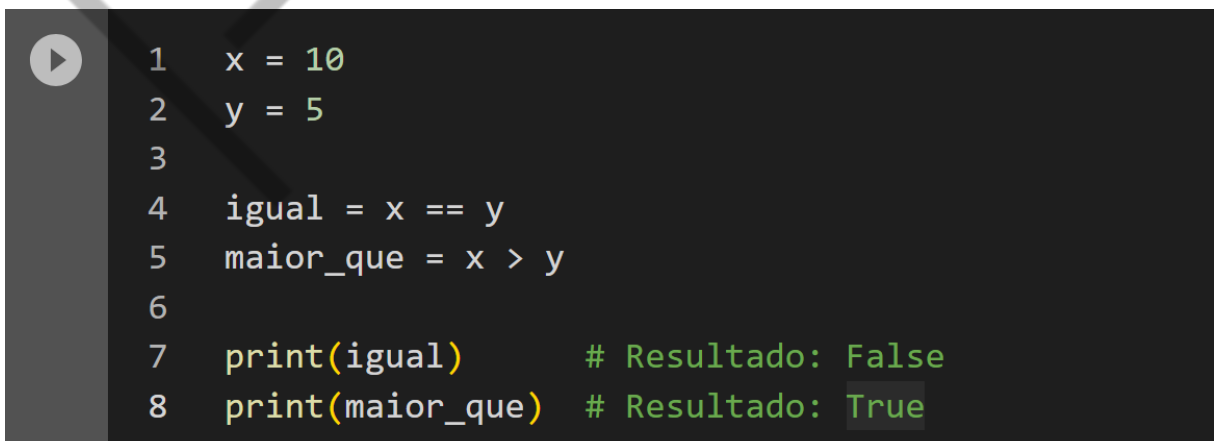
```
1  num1 = 5
2  num2 = 2
3
4  soma = num1 + num2
5  resto = num1 % num2
6
7  print(soma)  # Resultado: 7
8  print(resto) # Resultado: 1
```

Figura 9 – Exemplo de código com as operações soma e resto da divisão
Fonte: Elaborado pelo autor (2024)

Operadores de comparação:

- (==) igual à
- (!=) diferente de
- (<) menor que
- (>) maior que
- (<=) menor ou igual à
- (>=) maior ou igual à

Exemplo:



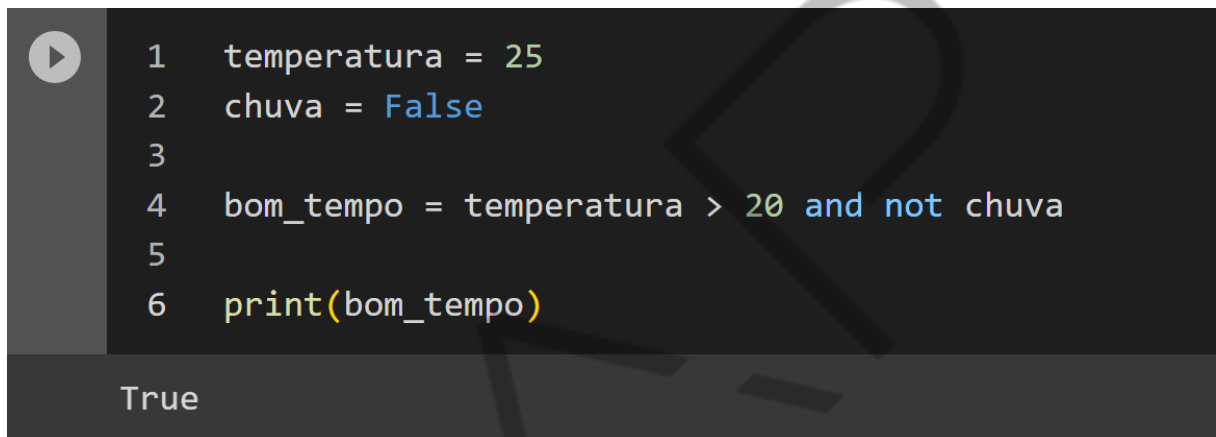
```
1  x = 10
2  y = 5
3
4  igual = x == y
5  maior_que = x > y
6
7  print(igual)      # Resultado: False
8  print(maior_que)  # Resultado: True
```

Figura 30 – Exemplo de código com as operações “igual a” e “maior que”
Fonte: Elaborado pelo autor (2024)

Operadores lógicos:

- (and) e lógico
- (or) ou lógico
- (not) não lógico

Exemplo:



```
1  temperatura = 25
2  chuva = False
3
4  bom_tempo = temperatura > 20 and not chuva
5
6  print(bom_tempo)
```

True

Figura 41 – Exemplo de código com as operações “maior que”, “and” e “not”.
Fonte: Elaborado pelo autor (2024)

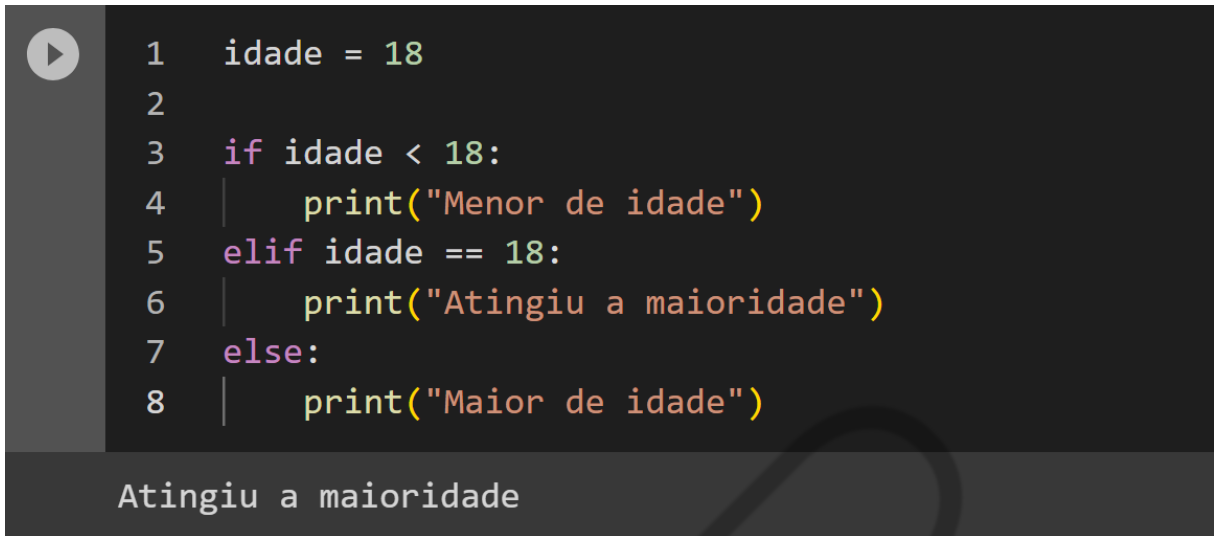
Estruturas de controle

Estas estruturas permitem controlar o fluxo de execução do programa.

Condicional:

- (if) se
- (elif) senão, se
- (else) senão

Exemplo:



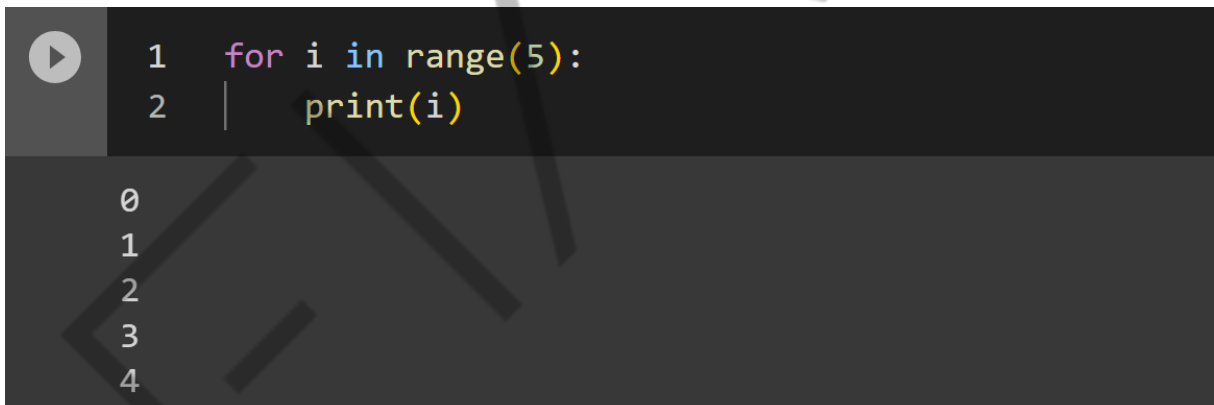
```
1  idade = 18
2
3  if idade < 18:
4      print("Menor de idade")
5  elif idade == 18:
6      print("Atingiu a maioridade")
7  else:
8      print("Maior de idade")
```

Atingiu a maioridade

Figura 52 – Exemplo de código com os condicionais “if”, “elif” e “else”
Fonte: Elaborado pelo autor (2024)

Estrutura de repetição (for):

Exemplo:



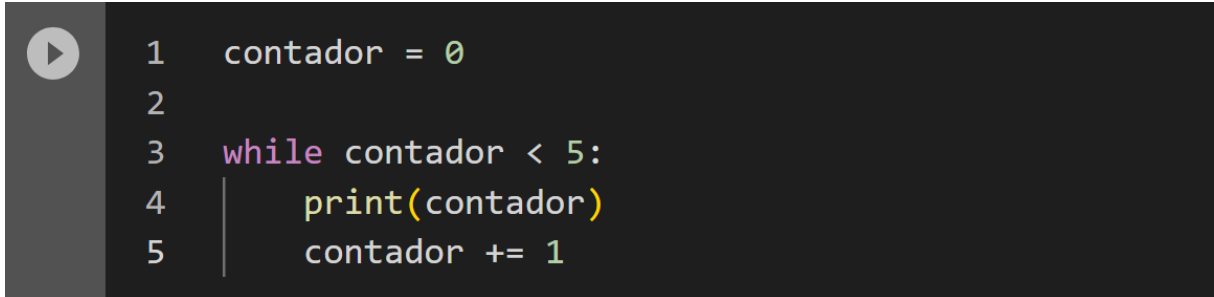
```
1  for i in range(5):
2      print(i)
```

0
1
2
3
4

Figura 63 – Exemplo de código com “for”
Fonte: Elaborado pelo autor (2024)

Estrutura de repetição (while):

Exemplo:



```
1 contador = 0
2
3 while contador < 5:
4     print(contador)
5     contador += 1
```

Figura 74 – Exemplo de código com “while”
Fonte: Elaborado pelo autor (2024)

Estrutura de controle (break | continue):

Exemplos:

break

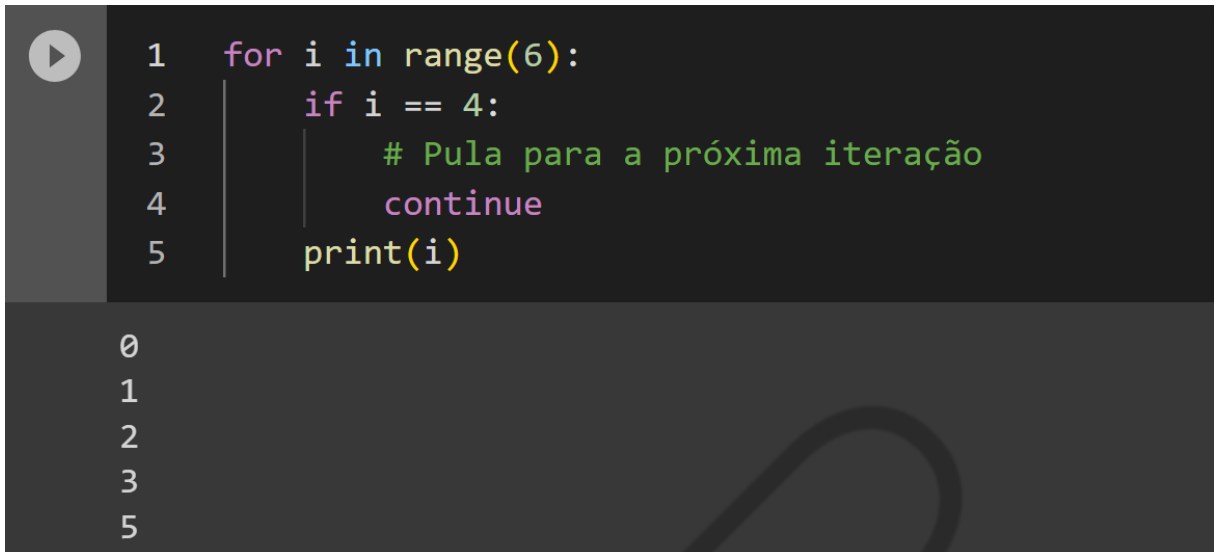


```
1 for i in range(10):
2     if i == 3:
3         break # Interrompe o loop
4     print(i)
```

0
1
2

Figura 85 – Exemplo de código com “for”, “if” e “break”
Fonte: Elaborado pelo autor (2024)

continue



```
1 for i in range(6):
2     if i == 4:
3         # Pula para a próxima iteração
4         continue
5     print(i)
```

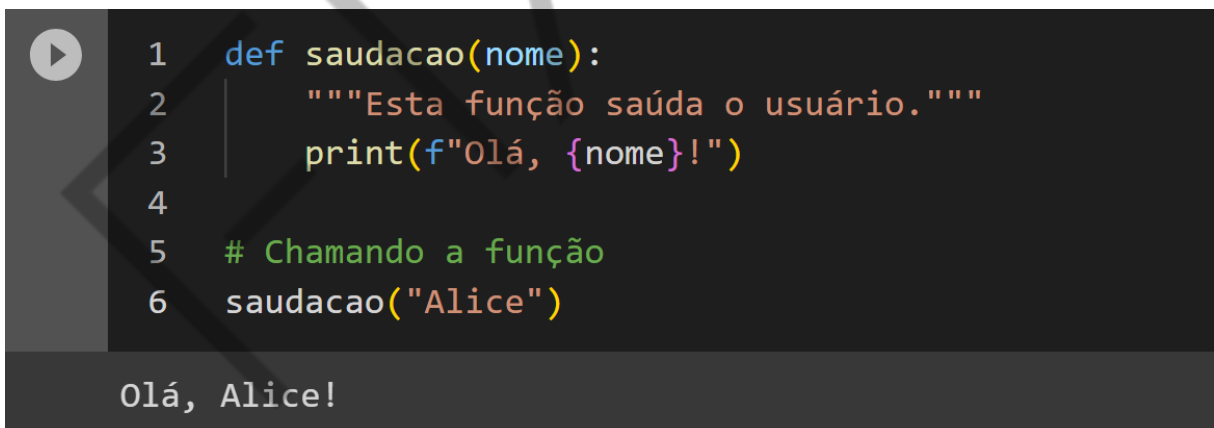
0
1
2
3
5

Figura 96 – Exemplo de código com “for”, “if” e “continue”
Fonte: Elaborado pelo autor (2024)

Funções

Uma função é um bloco de código reutilizável que realiza uma tarefa específica. As funções são definidas a partir da palavra-chave “def”.

Exemplo:



```
1 def saudacao(nome):
2     """Esta função saúda o usuário."""
3     print(f"Olá, {nome}!")
4
5 # Chamando a função
6 saudacao("Alice")
```

Olá, Alice!

Figura 107 – Exemplo de código com função “saudacao” definida a partir de “def”
Fonte: Elaborado pelo autor (2024)

Neste exemplo, a função “saudacao” recebe um argumento “nome” e imprime uma saudação personalizada.

Módulos

Um módulo é um arquivo contendo definições e instruções Python que podem ser reutilizadas em outros programas Python.

Tendo isso em mente, vamos ver como as funções podem ser usadas como módulos em um projeto Python? Analise o exemplo a seguir.

- **Contexto:** temos um texto sobre o porquê estudar programação e iremos aplicar um método criado para processar este texto a fim de retirar acentuações, possíveis caracteres especiais e convertê-lo integralmente para letras maiúsculas.
- **Justificativa:** o Python é uma linguagem de programação case sensitive, ou seja, faz distinção de letras maiúsculas e minúsculas. Desta forma, podemos ter dificuldades em encontrar, extrair e/ou alterar alguma “string” contida no texto. Assim, fazemos estas alterações mencionadas anteriormente e outras mais, caso haja necessidade, para tratar um texto e conseguir obter as informações necessárias.
- **Estrutura do projeto:**

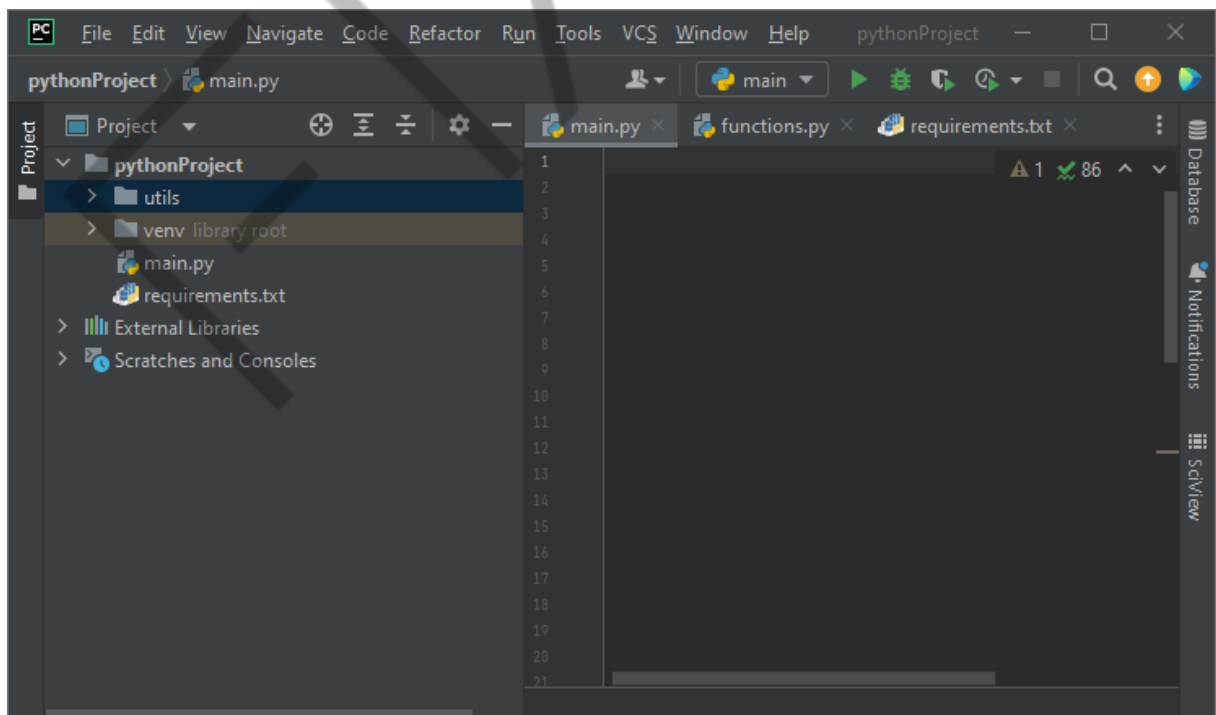
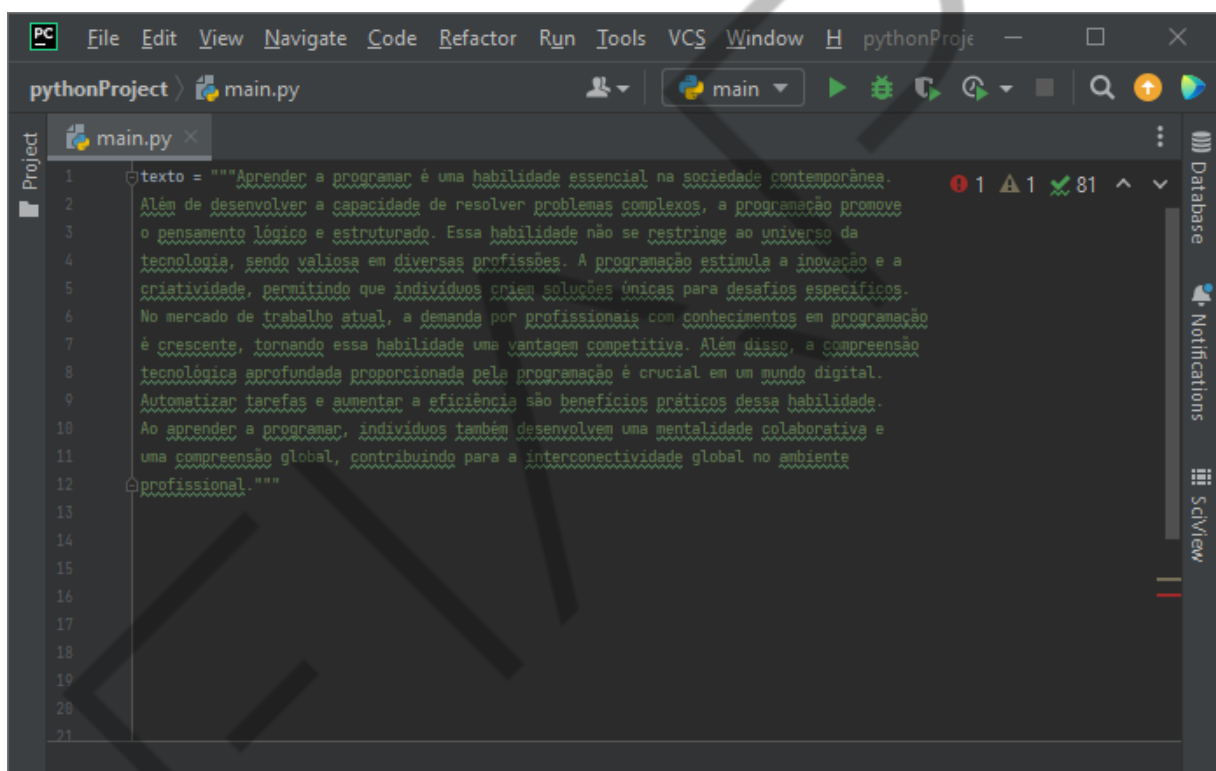


Figura 118 – Print de tela de um projeto Python
Fonte: Elaborado pelo autor (2024)

Na imagem anterior temos um projeto em Python criado com um arquivo “main.py”, que é nosso código principal, seguido do arquivo “requirements.txt”, que é responsável por receber as instalações necessárias para “rodar” o projeto.

Temos também dois diretórios, o “venv”, que se refere ao ambiente virtual em que serão instaladas as bibliotecas, e por fim o “utils”, onde encontraremos o arquivo “function.py”, no qual haverá todos os métodos criados para executar o programa e deixar o código mais legível.

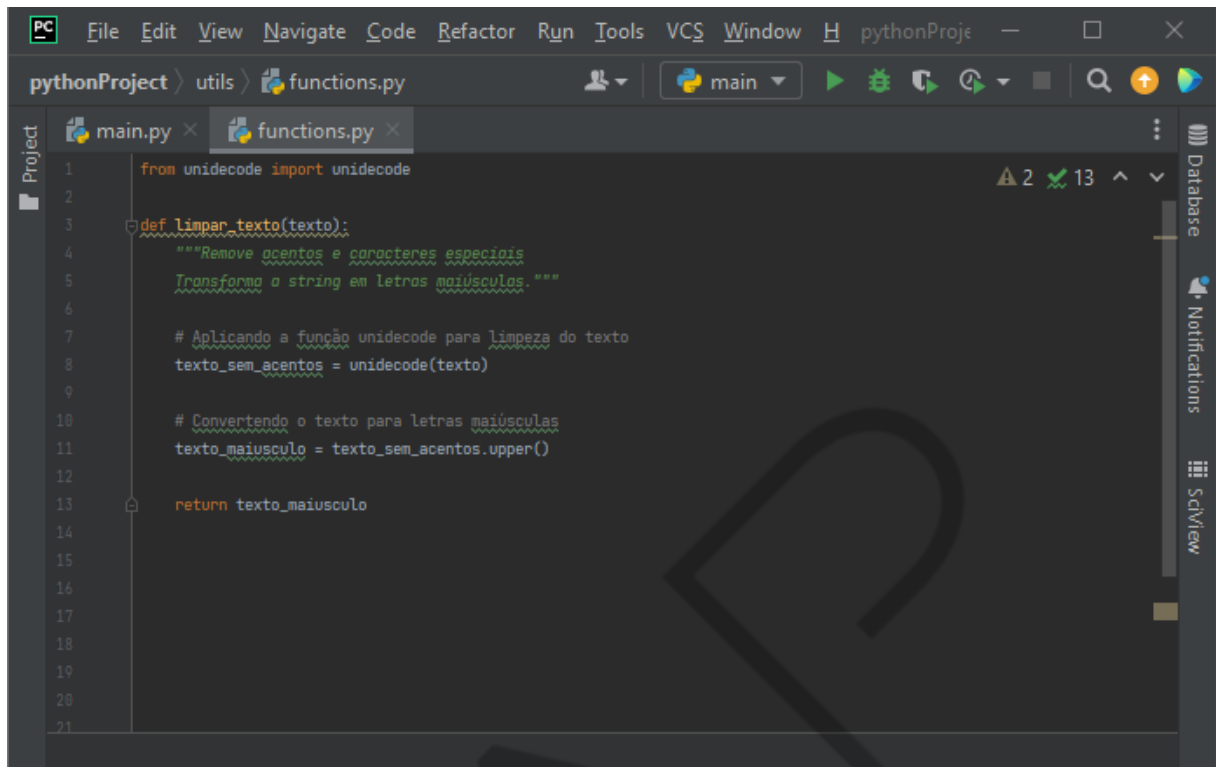


```
1 texto = """Aprender a programar é uma habilidade essencial na sociedade contemporânea.  
2 Além de desenvolver a capacidade de resolver problemas complexos, a programação promove  
3 o pensamento lógico e estruturado. Essa habilidade não se restringe ao universo da  
4 tecnologia, sendo valiosa em diversas profissões. A programação estimula a inovação e a  
5 criatividade, permitindo que indivíduos criem soluções únicas para desafios específicos.  
6 No mercado de trabalho atual, a demanda por profissionais com conhecimentos em programação  
7 é crescente, tornando essa habilidade uma vantagem competitiva. Além disso, a compreensão  
8 tecnológica aprofundada proporcionada pela programação é crucial em um mundo digital.  
9 Automatizar tarefas e aumentar a eficiência são benefícios práticos dessa habilidade.  
10 Ao aprender a programar, indivíduos também desenvolvem uma mentalidade colaborativa e  
11 uma compreensão global, contribuindo para a interconectividade global no ambiente  
12 profissional."""  
13  
14  
15  
16  
17  
18  
19  
20  
21
```

Figura 129 – Print de tela do arquivo “main.py”

Fonte: Elaborado pelo autor (2024)

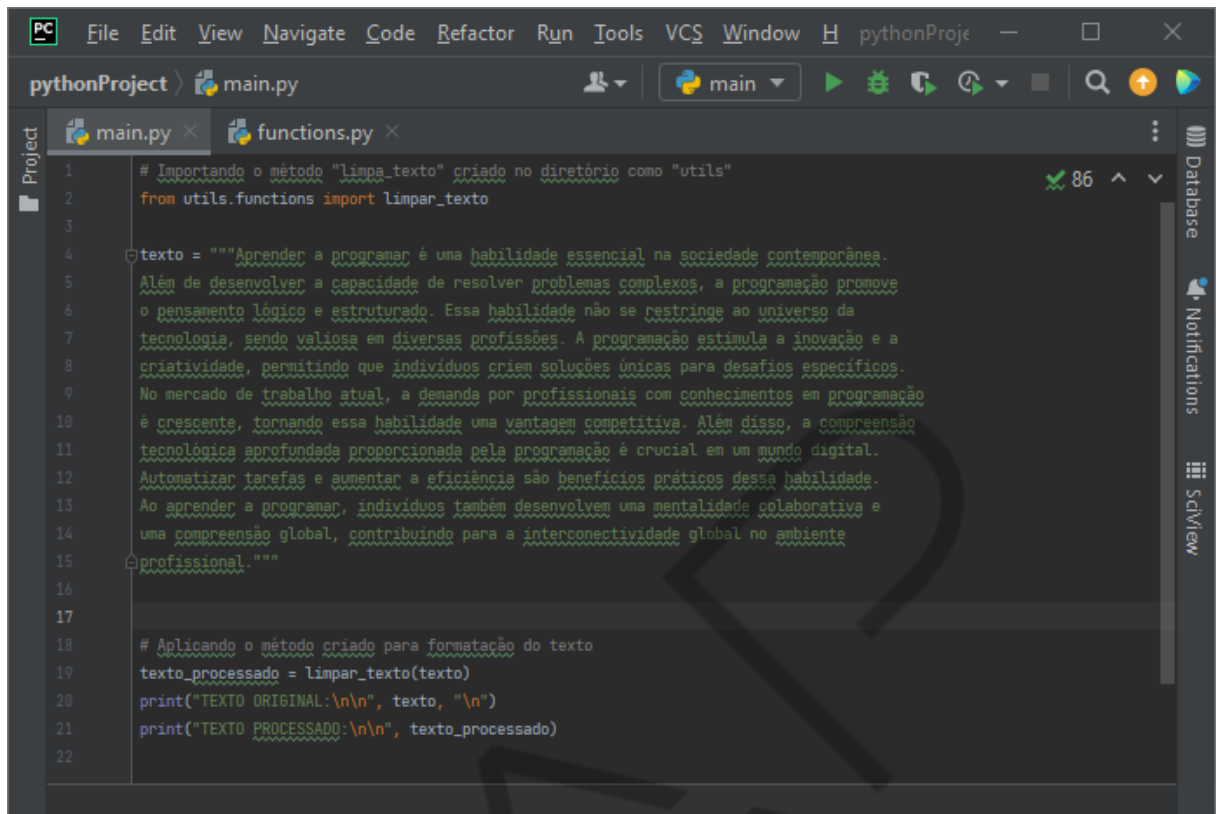
Nosso texto é uma string que está salva dentro da variável chamada de “texto”.

A screenshot of a code editor window titled 'pythonProject' showing the file 'utils/functions.py'. The code defines a function 'limpar_texto(texto)' that removes accents and converts text to uppercase using the 'unicode' module. The function is documented with a docstring in Portuguese. The editor interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help), a toolbar with icons for running, debugging, and other actions, and a sidebar on the left showing the project structure with 'main.py' and 'functions.py' tabs. The right sidebar contains panels for 'Database', 'Notifications', and 'ScView'. The code is as follows:

```
1 from unicode import unicode
2
3 def limpar_texto(texto):
4     """Remove acentos e caracteres especiais
5     Transforma a string em letras maiúsculas."""
6
7     # Aplicando a função unicode para limpeza do texto
8     texto_sem_acentos = unicode(texto)
9
10    # Convertendo o texto para letras maiúsculas
11    texto_maiusculo = texto_sem_acentos.upper()
12
13    return texto_maiusculo
14
15
16
17
18
19
20
21
```

Figura 20 – Print de tela do arquivo “functions.py”
Fonte: Elaborado pelo autor (2024)

Dentro do diretório “utils” temos o arquivo “function.py”. Dentro dele há a função (método) chamada “limpa_texto”, que realiza o processamento mencionado no contexto deste exemplo.



```
1 # Importando o método "limpar_texto" criado no diretório como "utils"
2 from utils.functions import limpar_texto
3
4 texto = """Aprender a programar é uma habilidade essencial na sociedade contemporânea.
5 Além de desenvolver a capacidade de resolver problemas complexos, a programação promove
6 o pensamento lógico e estruturado. Essa habilidade não se restringe ao universo da
7 tecnologia, sendo valiosa em diversas profissões. A programação estimula a inovação e a
8 criatividade, permitindo que indivíduos criem soluções únicas para desafios específicos.
9 No mercado de trabalho atual, a demanda por profissionais com conhecimentos em programação
10 é crescente, tornando essa habilidade uma vantagem competitiva. Além disso, a compreensão
11 tecnológica aprofundada proporcionada pela programação é crucial em um mundo digital.
12 Automatizar tarefas e aumentar a eficiência são benefícios práticos dessa habilidade.
13 Ao aprender a programar, indivíduos também desenvolvem uma mentalidade colaborativa e
14 uma compreensão global, contribuindo para a interconectividade global no ambiente
15 profissional."""
16
17
18 # Aplicando o método criado para formatação do texto
19 texto_processado = limpar_texto(texto)
20 print("TEXTO ORIGINAL:\n\n", texto, "\n")
21 print("TEXTO PROCESSADO:\n\n", texto_processado)
22
```

Figura 21 – Print de tela do arquivo “main.py” com a importação de “limpar_texto”
Fonte:

Na linha 2, estamos importando o método criado para o arquivo principal “main.py”; na linha 19, imprimimos o texto original; e na linha 20, o texto processado.

Texto original:

```
TEXTO ORIGINAL:

Aprender a programar é uma habilidade essencial na sociedade contemporânea.
Além de desenvolver a capacidade de resolver problemas complexos, a programação promove
o pensamento lógico e estruturado. Essa habilidade não se restringe ao universo da
tecnologia, sendo valiosa em diversas profissões. A programação estimula a inovação e a
criatividade, permitindo que indivíduos criem soluções únicas para desafios específicos.
No mercado de trabalho atual, a demanda por profissionais com conhecimentos em programação
é crescente, tornando essa habilidade uma vantagem competitiva. Além disso, a compreensão
tecnológica aprofundada proporcionada pela programação é crucial em um mundo digital.
Automatizar tarefas e aumentar a eficiência são benefícios práticos dessa habilidade.
Ao aprender a programar, indivíduos também desenvolvem uma mentalidade colaborativa e
uma compreensão global, contribuindo para a interconectividade global no ambiente
profissional.
```

Figura 22 – Print de tela com a impressão do texto original
Fonte: Elaborado pelo autor (2024)

Texto processado:

```
TEXTO PROCESSADO:

APRENDER A PROGRAMAR É UMA HABILIDADE ESSENCIAL NA SOCIEDADE CONTEMPORÂNEA.
ALÉM DE DESENVOLVER A CAPACIDADE DE RESOLVER PROBLEMAS COMPLEXOS, A PROGRAMAÇÃO PROMOVE
O PENSAMENTO LÓGICO E ESTRUTURADO. ESSA HABILIDADE NÃO SE RESTRINGE AO UNIVERSO DA
TECNOLOGIA, SENDO VALIOSA EM DIVERSAS PROFISSÕES. A PROGRAMAÇÃO ESTIMULA A INOVAÇÃO E A
CRIATIVIDADE, PERMITINDO QUE INDIVÍDUOS CRIEM SOLUÇÕES ÚNICAS PARA DESAFIOS ESPECÍFICOS.
NO MERCADO DE TRABALHO ATUAL, A DEMANDA POR PROFISSIONAIS COM CONHECIMENTOS EM PROGRAMAÇÃO
É CRESCENTE, TORNANDO ESSA HABILIDADE UMA VANTAGEM COMPETITIVA. ALÉM DISSO, A COMPREENSÃO
TECNOLOGICA APROFUNDADA PROPORCIONADA PELA PROGRAMAÇÃO É CRUCIAL EM UM MUNDO DIGITAL.
AUTOMATIZAR TAREFAS E AUMENTAR A EFICIÊNCIA SÃO BENEFÍCIOS PRÁTICOS DESSA HABILIDADE.
AO APRENDER A PROGRAMAR, INDIVÍDUOS TAMBÉM DESENVOLVEM UMA MENTALIDADE COLABORATIVA E
UMA COMPREENSÃO GLOBAL, CONTRIBUINDO PARA A INTERCONECTIVIDADE GLOBAL NO AMBIENTE
PROFISSIONAL.

Process finished with exit code 0
```

Figura 23 – Print de tela com a impressão do texto processado
Fonte: Elaborado pelo autor (2024)

Conclusão

Funções nativas do Python

Nem sempre precisamos “inventar a roda”: muitas vezes já existem soluções para nossos problemas, só precisamos pesquisar. Na documentação do Python, por exemplo, vocês irão encontrar uma infinidade de funções nativas que desempenham um papel crucial no nosso dia a dia.

Exemplos:

print(): esta função é usada para imprimir mensagens ou valores no console.

```
1 print("Olá, mundo!")

Olá, mundo!
```

Figura 24 – Exemplo de código com a função “print()”
Fonte: Elaborado pelo autor (2024)

len(): retorna o comprimento (número de elementos) de um objeto, como uma lista, tupla ou string.

```
[2] 1 my_list = [1, 2, 3, 4, 5]
    2 length = len(my_list)
    3 print(length) # Saída: 5
```

```
5
```

Figura 25 – Exemplo de código com as funções “len()” e “print()”
Fonte: Elaborado pelo autor (2024)

type(): retorna o tipo de um objeto.

```
1 x = 5
2 y = "Hello"
3 print(type(x)) # Saída: <class 'int'>
4 print(type(y)) # Saída: <class 'str'>
```

```
<class 'int'>
<class 'str'>
```

Figura 26 – Exemplo de código com as funções “print()” e “type()”
Fonte: Elaborado pelo autor (2024)

sorted(): retorna uma nova lista ordenada a partir dos elementos de uma sequência.

```
[8] 1 numeros = [3, 1, 4, 1, 5, 9, 2]
    2 numeros_ordenados = sorted(numeros)
    3 print(numeros_ordenados)
```

```
[1, 1, 2, 3, 4, 5, 9]
```

Figura 27 – Exemplo de código com as funções “sorted()” e “print()”
Fonte: Elaborado pelo autor (2024)

Chegou a hora da jornada de aprofundamento! Vamos explorar conceitos adicionais que enriquecerão sua compreensão dos fundamentos de Python. Agora, nos concentraremos nos conceitos fundamentais de Python I e falaremos

sobre tipos de dados, operadores de controle, funções e módulos. Iremos em nuances e aplicações práticas que irão ampliar suas habilidades.

Dicas

Antes de iniciar esta sessão, pratique os conceitos apresentados nesta aula, a teoria é valiosa e a prática é fundamental. Evite copiar códigos; em vez disso, compreenda cada linha. Esteja preparado(a) para assistir as aulas várias vezes, esclarecendo dúvidas à medida que surgem.

A seguir

Ao longo das próximas páginas, exploraremos tipos de dados avançados, estruturas de controle aprimoradas e conceitos avançados de programação funcional em Python. Vamos expandir seus conhecimentos e levar suas habilidades de programação ao próximo nível.

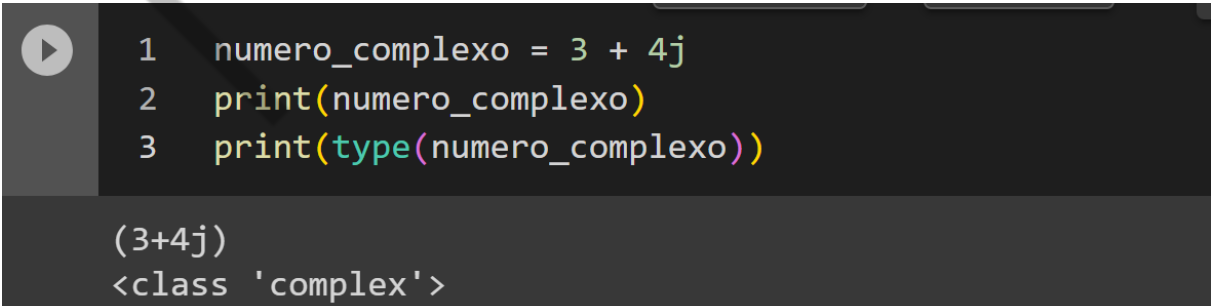
Explorando os tipos de dados avançados

Nesta seção, vamos conhecer dados pouco mencionados em cursos: a ideia é trazer para vocês conhecimentos agregadores que podem fornecer as ferramentas necessárias para se destacar no mercado de trabalho.

Números complexos

Além de números inteiros e ponto flutuante, o Python suporta os famosos números complexos.

Exemplo:



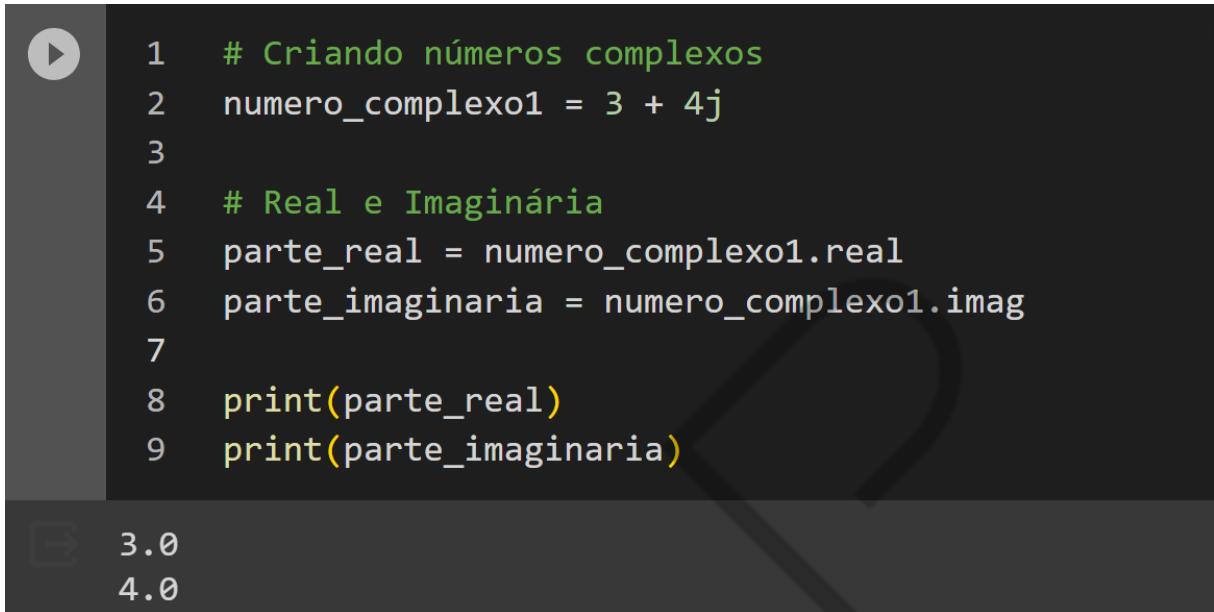
```
1  numero_complexo = 3 + 4j
2  print(numero_complexo)
3  print(type(numero_complexo))

(3+4j)
<class 'complex'>
```

Figura 28 – Exemplo de código com números complexos
Fonte: Elaborado pelo autor (2024)

A classe “complex” em Python é utilizada para lidar com números complexos. Estes possuem uma parte real e uma parte imaginária e são representados na forma

“ $a + bj$ ”, em que “a” é a parte real e “b” é a parte imaginária. Já o “j” é a unidade imaginária, ou seja: é a raiz quadrada de (-1).

A screenshot of a Python code editor with a dark background. The code is as follows:

```
1  # Criando números complexos
2  numero_complexo1 = 3 + 4j
3
4  # Real e Imaginária
5  parte_real = numero_complexo1.real
6  parte_imaginaria = numero_complexo1.imag
7
8  print(parte_real)
9  print(parte_imaginaria)
```

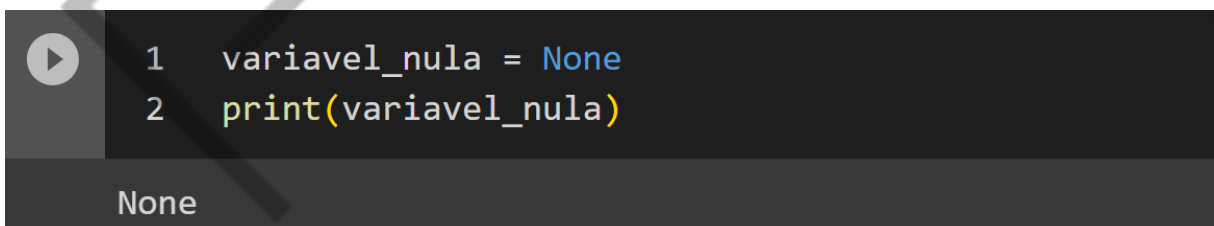
Below the code, the output is displayed: 3.0 and 4.0 on separate lines.

Figura 29 – Exemplo de código com números complexos (2)
Fonte: Elaborado pelo autor (2024)

Dados none

Representa o valor nulo ou ausência de valor; é útil em situações em que seja interessante indicar que o valor é inválido.

Exemplo:

A screenshot of a Python code editor with a dark background. The code is as follows:

```
1  variavel_nula = None
2  print(variavel_nula)
```

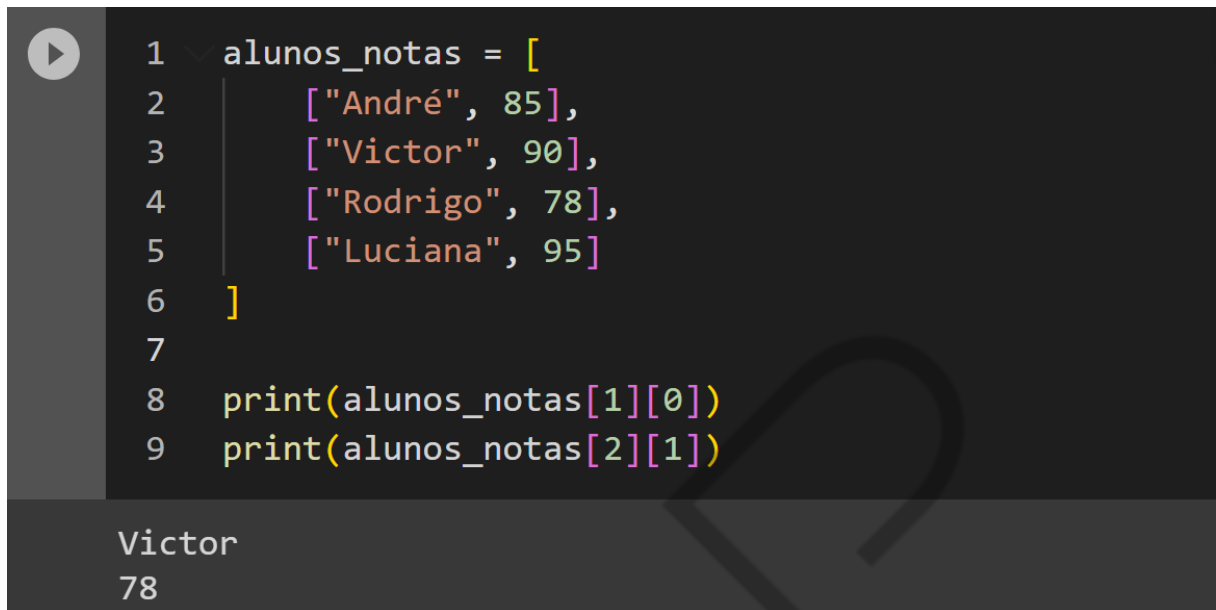
Below the code, the output is displayed: None.

Figura 30 – Exemplo de código com “None”
Fonte: Elaborado pelo autor (2024)

Listas Aninhadas

Listas aninhadas em Python referem-se ao conceito de incluir uma lista dentro de outra lista. Isso permite criar estruturas de dados mais complexas, em que cada elemento da lista externa é, na verdade, uma lista interna. Com listas aninhadas, você pode criar matrizes, tabelas ou estruturas de dados multidimensionais.

Exemplo:



```
1  alunos_notas = [  
2      ["André", 85],  
3      ["Victor", 90],  
4      ["Rodrigo", 78],  
5      ["Luciana", 95]  
6  ]  
7  
8  print(alunos_notas[1][0])  
9  print(alunos_notas[2][1])
```

Victor
78

Figura 31 – Exemplo de código com lista aninhada
Fonte: Elaborado pelo autor (2024)

Compreensão de Listas

A compreensão de listas ou “list comprehension” em inglês é uma maneira concisa e expressiva de criar listas em Python; muitos(as) profissionais mencionam que “codar” usando “list comprehension” é um jeito “pythônico”. Esta estrutura permite criar uma lista concomitante a um loop usando apenas uma linha.

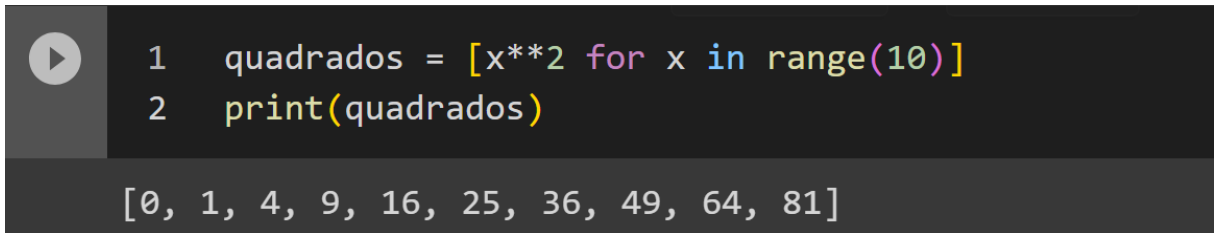
A estrutura básica de uma compreensão de listas é a seguinte:

[expressao for item in iteravel]

Em que:

- “expressão” é o valor a ser incluído na lista.
- “item” é uma variável que assume o valor do iterável.
- “iterável” é a sequência de elementos a serem percorridos.

Exemplo:



```
1 quadrados = [x**2 for x in range(10)]
2 print(quadrados)
```

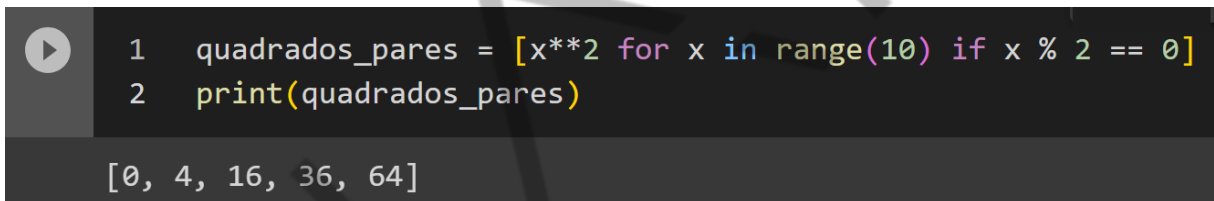
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Figura 32 – Exemplo de código com compreensão de lista
Fonte: Elaborado pelo autor (2024)

Neste exemplo, a compreensão de lista “x**2” é avaliada para cada valor de “x” no intervalo de 0 a 9, criando assim uma lista dos quadrados dos números de 0 a 9.

Você também pode adicionar uma cláusula “if” para filtrar elementos. Por exemplo, aqui está uma compreensão de lista que cria uma lista de quadrados apenas para os números pares de 0 a 9:

Exemplo:



```
1 quadrados_pares = [x**2 for x in range(10) if x % 2 == 0]
2 print(quadrados_pares)
```

[0, 4, 16, 36, 64]

Figura 33 – Exemplo de código com compreensão de lista e com filtro “if”.
Fonte: Elaborado pelo autor (2024)

Essa compreensão de lista utiliza a cláusula “if x % 2 == 0” para incluir apenas os números pares na lista resultante.

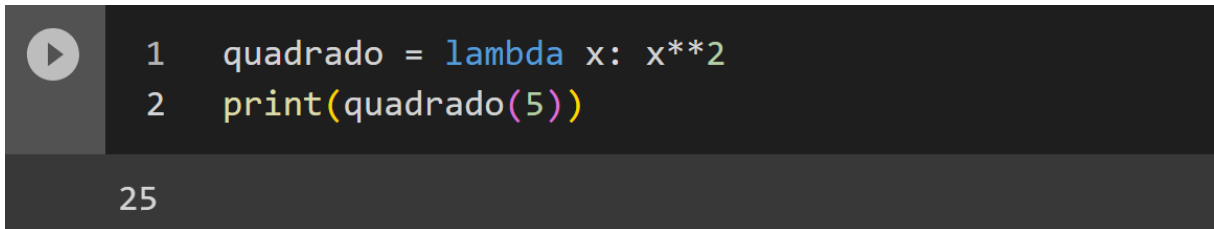
As compreensões de lista proporcionam uma maneira eficiente e legível de criar listas em Python, reduzindo a necessidade de escrever loops tradicionais para construir listas.

Função lambda

Em Python, uma função lambda é uma forma de criar funções anônimas e pequenas usando a palavra-chave “lambda”. Essas funções são úteis quando você precisa de uma função por um curto período e não deseja definir uma função completa usando a declaração def. A sintaxe básica de uma função lambda é:

Lambda argumento: expressao

Exemplo:



```
1  quadrado = lambda x: x**2
2  print(quadrado(5))
```

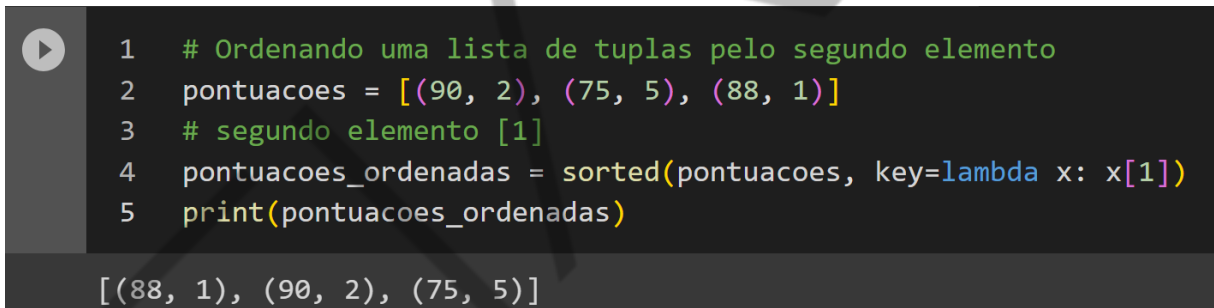
25

Figura 34 – Exemplo de código com “lambda”
Fonte: Elaborado pelo autor (2024)

Neste exemplo, lambda “x: x**2” cria uma função que aceita um argumento “x” e retorna o quadrado desse argumento. A função lambda é atribuída à variável quadrado e podemos chamá-la como qualquer outra função.

As funções lambda são frequentemente usadas em situações nas quais funções pequenas são necessárias, como em operações de ordenação, filtragem ou mapeamento de listas.

Exemplo:



```
1  # Ordenando uma lista de tuplas pelo segundo elemento
2  pontuacoes = [(90, 2), (75, 5), (88, 1)]
3  # segundo elemento [1]
4  pontuacoes_ordenadas = sorted(pontuacoes, key=lambda x: x[1])
5  print(pontuacoes_ordenadas)
```

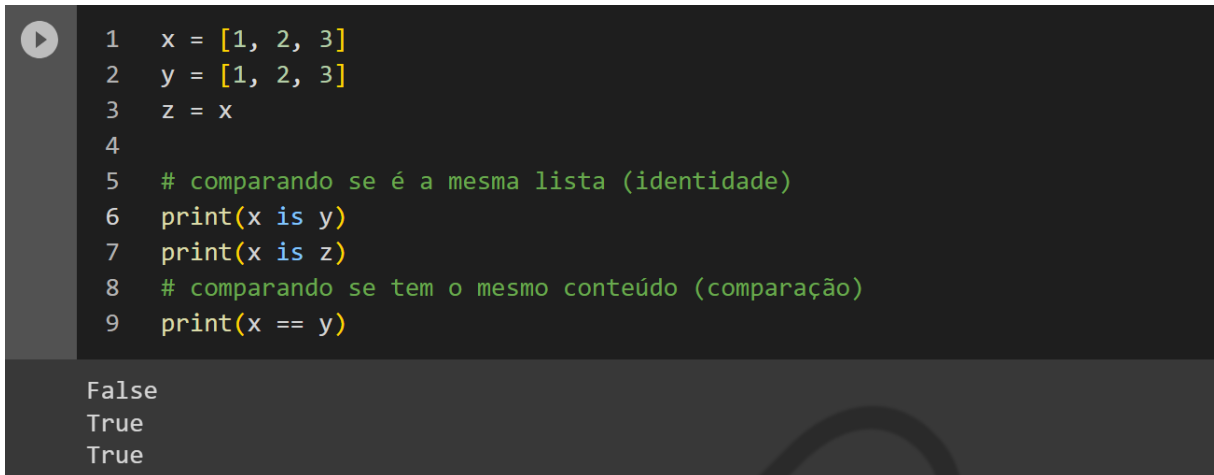
[(88, 1), (90, 2), (75, 5)]

Figura 35 – Exemplo de código com “sorted()” e “lambda”
Fonte: Elaborado pelo autor (2024)

Operadores de identidade

Além dos operadores de comparação, temos também os operadores de identidade “is” e “is not” (inverso de “is”), que são usados para verificar se dois objetos são ou não o mesmo objeto.

Exemplo:



```
1 x = [1, 2, 3]
2 y = [1, 2, 3]
3 z = x
4
5 # comparando se é a mesma lista (identidade)
6 print(x is y)
7 print(x is z)
8 # comparando se tem o mesmo conteúdo (comparação)
9 print(x == y)
```

False
True
True

Figura 36 – Exemplo de código com o operador “is”
Fonte: Elaborado pelo autor (2024)

Note a diferença de um operador de identidade e um de comparação: o operador de identidade verifica se trata da mesma lista (identidade da lista) e o operador de comparação se trata do mesmo conteúdo.

Operadores de Atribuição Avançada

Os operadores de atribuição avançada em Python são versões compactas de operadores de atribuição que combinam uma operação com uma atribuição. Eles permitem que você realize uma operação e atribua o resultado a uma variável em uma única expressão, economizando código e tornando-o mais conciso.

Exemplo:

Operador aritmético

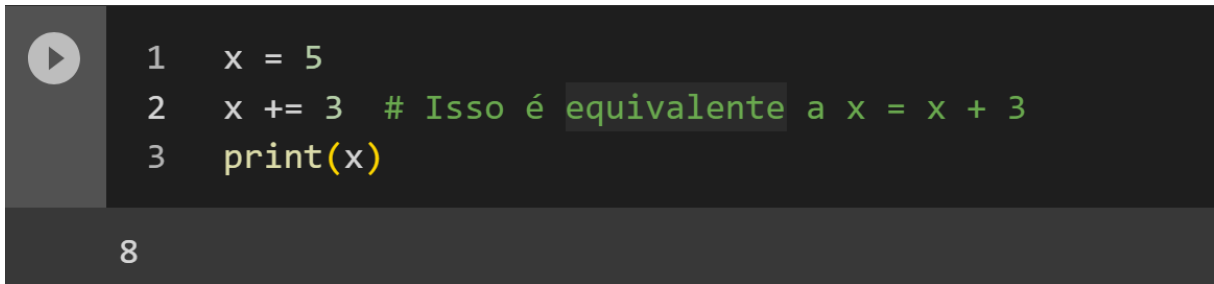


```
1 x = 5
2 x = x + 3
3 print(x)
```

8

Figura 37 – Exemplo de código com o operador aritmético soma
Fonte: Elaborado pelo autor (2024)

Operador de atribuição avançada



```
1 x = 5
2 x += 3 # Isso é equivalente a x = x + 3
3 print(x)
```

8

Figura 38 – Exemplo de código com o operador de atribuição avançada “+=”

Fonte: Elaborado pelo autor (2024)

Continue aprimorando suas habilidades com operadores. Na próxima aula teremos outras estruturas que podem lhe garantir a diferenciação de mercado.

MERCADO, CASES E TENDÊNCIAS

O deep learning é uma das tecnologias mais avançadas e promissoras no campo da inteligência artificial. Ele consiste em construir redes neurais artificiais que imitam o funcionamento do cérebro humano, permitindo que as máquinas aprendam a partir de grandes quantidades de dados e realizem tarefas complexas, como reconhecimento de imagem, processamento de linguagem natural, geração de texto e detecção de anomalias, entre outras.

O mercado de deep learning tem crescido rapidamente nos últimos anos, impulsionado pelo aumento da disponibilidade de dados, do poder computacional e dos avanços nos algoritmos. Segundo um relatório da Mordor Intelligence [s.d.], o mercado de deep learning deve registrar um CAGR de 42,56% durante o período de previsão de 2023 a 2028, alcançando um valor de US\$ 26,64 bilhões em 2028.

Os principais fatores que impulsionam esse crescimento são a melhoria da qualidade dos produtos e serviços, a otimização dos processos e fluxos de trabalho, a personalização da experiência do cliente, a inovação em diversos setores e a crescente demanda por soluções de análise preditiva,

O deep learning tem sido aplicado com sucesso em diversas indústrias e domínios, gerando benefícios e vantagens competitivas para as organizações que o adotam. Alguns exemplos de casos de uso de deep learning são:

Finanças

Pode ser usado para analisar dados financeiros, detectar fraudes, prever movimentos do mercado, otimizar portfólios e gerar relatórios, entre outras aplicações.

Um exemplo é o caso da empresa de investimentos Numerai, que usa uma rede neural criptografada para combinar as previsões de milhares de cientistas de dados anônimos(as) e criar um fundo de hedge meta-modelado.

Outro exemplo é o caso da empresa de pagamentos PayPal, que usa um sistema de deep learning para detectar transações fraudulentas em tempo real, reduzindo as perdas e melhorando a segurança de clientes.

Saúde

Pode ser usado para diagnosticar doenças, analisar imagens médicas, descobrir novos medicamentos, monitorar pacientes e auxiliar cirurgias, entre outras aplicações.

Um exemplo é o caso da empresa de biotecnologia Deep Genomics, que usa uma plataforma de deep learning para identificar mutações genéticas que causam doenças raras e desenvolver terapias personalizadas.

Outro exemplo é o caso da empresa de saúde Babylon Health, que usa um chatbot de deep learning para fornecer consultas médicas online, triagem de sintomas e encaminhamento para especialistas.

Varejo

Pode ser usado para recomendar produtos, segmentar clientes, prever demanda, gerenciar estoque e otimizar preços, entre outras aplicações.

Um exemplo é o caso da empresa de comércio eletrônico Amazon, que usa um sistema de deep learning para gerar descrições de produtos, classificar imagens, traduzir idiomas, reconhecer voz e fornecer assistência virtual.

Outro exemplo é o caso da empresa de moda Stitch Fix, que usa um sistema de deep learning para criar estilos personalizados, selecionar roupas, enviar caixas de assinatura e obter feedback de clientes.

Automotivo

Pode ser usado para desenvolver carros autônomos, melhorar a segurança e otimizar o consumo de combustível, entre outras aplicações.

Um exemplo é o caso da empresa de tecnologia Tesla, que usa uma rede neural de deep learning para processar as imagens capturadas pelas câmeras dos carros e controlar a direção, a aceleração, a frenagem e a navegação.

Outro exemplo é o caso da empresa de mobilidade Uber, que usa um sistema de deep learning para estimar o tempo de chegada, calcular a tarifa, rotear motoristas, detectar incidentes e fornecer suporte ao cliente.

Telecomunicações e mídia

Pode ser usado para melhorar a qualidade do sinal, otimizar a rede, gerar conteúdo, personalizar a publicidade, entre outras aplicações.

Um exemplo é o caso da empresa de telecomunicações Verizon, que usa um sistema de deep learning para analisar o tráfego de dados, prever falhas, solucionar problemas e melhorar a experiência de usuário.

Outro exemplo é o caso da empresa de entretenimento Netflix, que usa um sistema de deep learning para recomendar filmes e séries, otimizar a transmissão, gerar legendas, criar trailers e avaliar o desempenho do conteúdo.

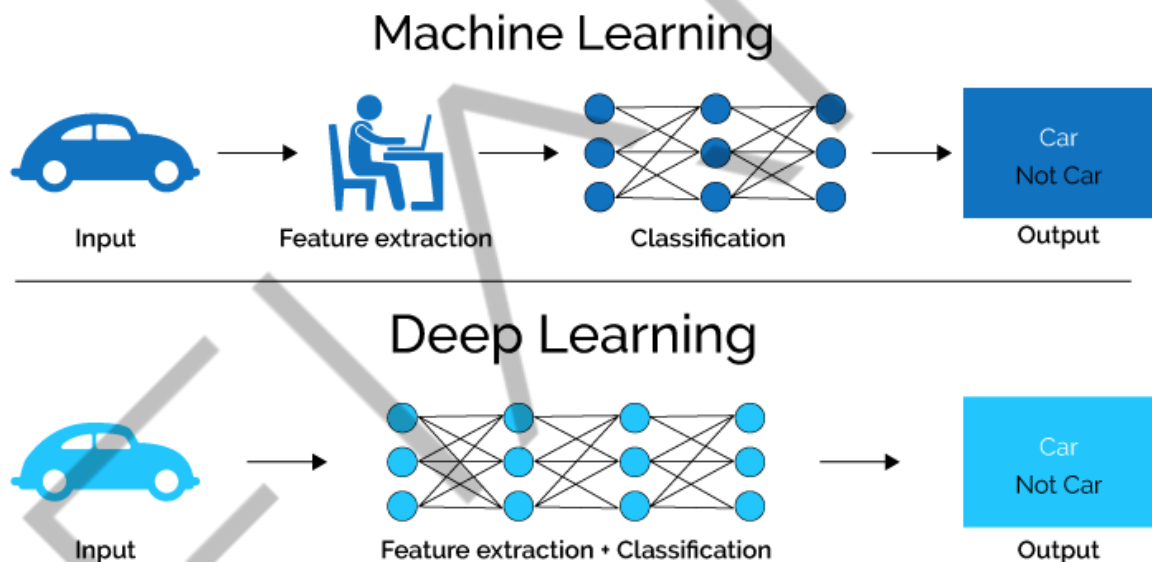


Figura 39 – Aprendizado de máquina X aprendizado profundo
Fonte: Software Testing Help (2024)

Histórias de Sucesso de Deep Learning

- **AlphaGo:** é um programa de computador que usa deep learning para jogar o jogo de tabuleiro Go, considerado um dos mais complexos e desafiadores do mundo. Ele foi desenvolvido pela empresa DeepMind, que faz parte do grupo Google. Em 2016, ele surpreendeu o mundo ao derrotar o campeão mundial

Lee Sedol por 4 a 1, demonstrando um nível de inteligência e criatividade superior ao dos humanos. Em 2017, ele se aposentou invicto, após vencer o campeão mundial Ke Jie por 3 a 0. O AlphaGo é considerado um marco na história da inteligência artificial e do deep learning.

- **FaceApp:** é um aplicativo de smartphone que usa deep learning para alterar as características faciais das pessoas, como idade, gênero, cabelo e sorriso, entre outras. Ele foi desenvolvido pela empresa Wireless Lab, sediada na Rússia. Em 2019, ele se tornou viral ao permitir que usuários vissem como seriam no futuro, gerando milhões de fotos e compartilhamentos nas redes sociais. O FaceApp usa redes neurais generativas adversariais (GANs) para criar imagens realistas e convincentes, baseadas nas fotos originais dos usuários. O FaceApp é considerado um exemplo de como o deep learning pode ser usado para fins de entretenimento e diversão.
- **OpenAI Codex:** é um sistema de deep learning que usa processamento de linguagem natural para gerar código de programação a partir de descrições em linguagem natural. Ele foi desenvolvido pela empresa OpenAI, que é uma organização de pesquisa em inteligência artificial. Em 2021 ele foi lançado como uma ferramenta online chamada Copilot em parceria com a empresa GitHub, que é uma plataforma de hospedagem de código. O OpenAI Codex usa uma rede neural recorrente (RNN) para aprender a partir de bilhões de linhas de código público e produzir código em mais de uma dúzia de linguagens, como Python, Java e C#, entre outras. O OpenAI Codex é considerado um exemplo de como o deep learning pode ser usado para fins de produtividade e inovação.

Tendências do Deep Learning

O deep learning é uma tecnologia em constante evolução que apresenta novas tendências e desafios para o futuro. Algumas das tendências que podem ser observadas são:

Aumento da complexidade e da diversidade das redes neurais

As redes neurais de deep learning estão se tornando cada vez mais complexas e diversificadas, incorporando novas arquiteturas, funções, camadas e técnicas para melhorar o desempenho e a eficiência.

Alguns exemplos de novas redes neurais são as redes neurais convolucionais (CNNs), as redes neurais recorrentes (RNNs), as redes neurais generativas adversariais (GANs), as redes neurais de grafos (GNNs) e as redes neurais de cápsulas (CapsNets), entre outras.

Integração com outras tecnologias

O deep learning está se integrando com outras tecnologias, como a computação em nuvem, a internet das coisas, a realidade aumentada, a realidade virtual e a blockchain, entre outras, para criar soluções mais completas e inovadoras.

Alguns exemplos de integração são o uso de nuvens de deep learning para fornecer serviços de inteligência artificial sob demanda, o uso de sensores de internet das coisas para coletar e processar dados para o deep learning, o uso de realidade aumentada e virtual para criar ambientes imersivos para o deep learning e o uso de blockchain para garantir a segurança e a transparência dos dados do deep learning, entre outros.

Expansão para novos domínios e aplicações

O deep learning está se expandindo para novos domínios e aplicações que antes eram considerados difíceis ou impossíveis de serem realizados por máquinas.

Alguns exemplos de novos domínios e aplicações são a arte, a música, a literatura, a educação, a psicologia, a filosofia e a ética. O deep learning pode ser usado para criar obras de arte, compor músicas, escrever textos, ensinar alunos, analisar emoções, debater ideias e resolver dilemas, entre outros.

O deep learning é uma tecnologia revolucionária, que está transformando indústrias e vidas. Ele oferece oportunidades e desafios para profissionais, organizações e a sociedade. Para acompanhar essa evolução, é preciso se atualizar, capacitar e preparar para o futuro.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, conhecemos um pouco dos conceitos fundamentais do Python e sua aplicação prática. Ainda estamos em estágios iniciais; portanto, treinem as habilidades adquiridas, pois só assim absorveremos ao máximo todas as informações: a cada aula aumenta consideravelmente o volume de informações e o grau de complexidade. Não deixem virar uma “bola de neve”.

Até a próxima!



REFERÊNCIAS

BORGES, L. E. **Python para Desenvolvedores**. [s.l.]: Alta Books, 2014.

FACEAPP. **FaceApp**. 2024. Disponível em: <<https://www.faceapp.com>>. Acesso em: 17 jan. 2024.

GOOGLE DEEPMIND. **AlphaGo**. 2024. Disponível em: <<https://deepmind.google/technologies/alphago/>>. Acesso em: 17 jan. 2024.

JOSHI, P. **Inteligência Artificial com Python: Uma Guia Prático**. [s.l.]: Novatec, 2018.

LUTZ, M. **Aprendendo Python**. [s.l.]: Novatec, 2014.

MCKINNEY, W. **Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython**. [s.l.]: O'Reilly Media, 2017.

MORDOR INTELLIGENCE. **Tamanho do mercado de aprendizagem profunda e análise de ações – Tendências e previsões de crescimento (2023 – 2028)**. [s.d.] Disponível em: <<https://www.mordorintelligence.com/pt/industry-reports/deep-learning>>. Acesso em: 17 jan. 2024.

OPENAI. **OpenAI Codex**. 2024. Disponível em: <<https://openai.com/blog/openai-codex>>. Acesso em: 17 jan. 2024.

SANTOS, L. **Deep Learning: o que é e como está transformando indústrias e vidas?** 2022. Disponível em: <<https://olhardigital.com.br/2022/05/09/colunistas/deep-learning-o-que-e-e-como-esta-transformando-industrias-e-vidas/>>. Acesso em: 17 jan. 2024.

PALAVRAS-CHAVE

Palavras-chave: Python. Programação. Deep Learning.

EMENDAS



POSTECH