Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Estrutura de controle de fluxo

Compreensão de listas

Aula 1

Código da aula: [DADOS]ANO1C2B2S13A1

Exposição

Objetivos da Aula

Revisar o conceito de listas e introduzir o conceito de compreensão de listas.

Competências da Unidade (Técnicas e Socioemocionais)

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados.
- Usar técnicas para explorar e analisar dados, aplicar modelos estatísticos, identificar padrões, realizar inferências e tomar decisões fundamentadas em evidências.
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados; trabalhar em equipes multifuncionais, colaborando com colegas, gestores e clientes.

Recursos Didáticos

- Recurso audiovisual para exibição de vídeos e imagens.
- Acesso ao laboratório de informática e/ou à internet.
- Software Anaconda/Jupyter Notebook instalados ou similar.

Duração da Aula

50 minutos

Motivação: transação financeira

Você trabalha em uma empresa financeira e observou o código de um programador que criou um programa em Python para analisar as transações financeiras.

Sabe-se que transações positivas são depósitos (valores positivos) e valores negativos são retiradas de dinheiro.

Analise o código no próximo slide e responda:

1. Você sabe se o programador fez o código de acordo com o problema pedido?

2. O que você identificou nas linhas 2 e 8?

Momento de **reflexão**

Exposição

Motivação: transação financeira

```python
def calcular_total_transacoes_positivas(transacoes):
    transacoes_positivas = [valor for valor in transacoes if valor > 0]
    total_transacoes_positivas = sum(transacoes_positivas)
    quantidade_transacoes_positivas = len(transacoes_positivas)
    return total_transacoes_positivas, quantidade_transacoes_positivas

def calcular_total_transacoes_negativas(transacoes):
    transacoes_negativas = [valor for valor in transacoes if valor < 0]
    total_transacoes_negativas = sum(transacoes_negativas)
    return total_transacoes_negativas

# Lista de transações financeiras de um cliente
transacoes = [100, -50, 200, -20, 150, -30, 180]

# Chamando as funções para calcular os totais
total_positivas, quantidade_positivas = calcular_total_transacoes_positivas(transacoes)
total_negativas = calcular_total_transacoes_negativas(transacoes)

# Impressão dos resultados
print("Total das transações positivas:", total_positivas)
print("Quantidade de transações positivas:", quantidade_positivas)
print("Total das transações negativas:", total_negativas)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

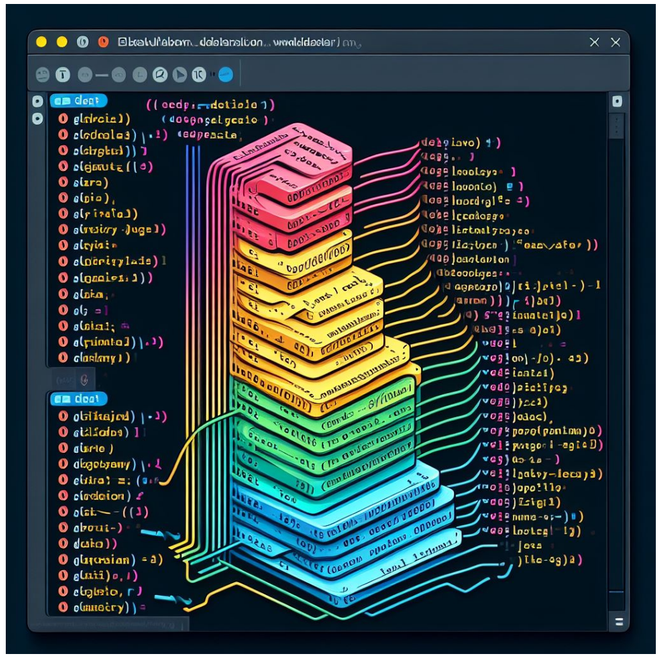
Exposição

Relembrando

	Listas
Definição	• Listas são estruturas de dados em Python que permitem armazenar uma coleção ordenada de elementos. • São mutáveis, o que significa que você pode adicionar, remover ou modificar elementos.
Sintaxe básica	• Declaradas usando colchetes []. • Exemplo: lista = [1, 2, 3, 4, 5].
Acesso aos elementos	• Índices começam em 0. • Exemplo: primeiro_elemento = lista[0].
Operações comuns	• Adição de elementos: lista.append(valor). • Remoção de elementos: lista.remove(valor) ou del lista[indice]. • Tamanho da lista: len(lista).
Versatilidade	• Aceita diferentes tipos de dados. • Pode conter listas dentro de listas (listas aninhadas).
Iteração e manipulação	• Fornece métodos poderosos para percorrer e manipular elementos. • Útil em várias operações, como filtragem, transformação e redução de dados.

Exposição

Compreensão de listas



Compreensão de lista ou ***List Comprehension*** em Python é uma construção sintática que permite criar listas de maneira concisa e eficiente.

Curiosidade:

Essa técnica é uma forma compacta de gerar, filtrar e transformar elementos de uma lista.

Elaborado especialmente para o curso com a ferramenta Microsoft Copilot.

Exposição

Compreensão de listas

Sua sintaxe básica é:

```
[expressao for elemento in lista_original if condicao]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Tome nota:

- **expressao:** A expressão a ser aplicada a cada elemento da lista.
- **elemento:** A variável que representa cada elemento da lista original.
- **lista_original:** A lista da qual os elementos são retirados.
- **condicao (opcional):** Uma condição que determina se o elemento será incluído na nova lista.

Exposição

Exemplo:

1. Criação simples

```
1  quadrados = [x**2 for x in range(1, 6)]
2
3  print(quadrados)
```

```
[1, 4, 9, 16, 25]
```

```
[expressao for elemento in lista_original if condicao]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Tome nota:

expressao: x**2 – A expressão que eleva cada elemento x ao quadrado.

elemento: for x in range(1, 6) - A variável x representa cada elemento na lista original, que varia de 1 a 5.

lista_original: range(1, 6) – A lista original da qual os elementos são retirados.

condicao: Não há condição neste exemplo.

Exposição

Exemplo:

Vamos reescrever o código e comparar.

```
1  quadrados = [x**2 for x in range(1, 6)]
2
3  print(quadrados)
```

```
[1, 4, 9, 16, 25]
```

```
1  quadrados = []
2  for x in range(1, 6):
3      quadrados.append(x**2)
4
5  print(quadrados)
```

```
[1, 4, 9, 16, 25]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Exemplo:

2. Filtragem de elementos

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

```
[0, 2, 4, 6, 8]
```

```
[expressao for elemento in lista_original if condicao]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Tome nota:

expressao: x – A expressão que representa cada elemento x.

elemento: for x in range(10) – A variável x representa cada elemento na lista original, variando de 0 a 9.

lista_original: range(10) – A lista original da qual os elementos são retirados.

condicao: if x % 2 == 0 – A condição que verifica se o número é par.

Exposição

Exemplo:

Vamos reescrever o código e comparar:

```python
numeros_pares = [x for x in range(10) if x % 2 == 0]
print(numeros_pares)
```

```
[0, 2, 4, 6, 8]
```

```python
numeros_pares = []
for x in range(10):
    if x % 2 == 0:
        numeros_pares.append(x)

print(numeros_pares)
```

```
[0, 2, 4, 6, 8]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Exemplo:

3. Transformação de elementos

```python
palavras = ['python', 'é', 'incrível']
maiusculas = [palavra.upper() for palavra in palavras]

print(maiusculas)
```

```
['PYTHON', 'É', 'INCRÍVEL']
```

```
[expressao for elemento in lista_original if condicao]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Tome nota:

expressao: palavra.upper() – A expressão que converte cada palavra para maiúsculas.

elemento: for palavra in palavras – A variável palavra representa cada elemento na lista original palavras.

lista_original: palavras – A lista original da qual os elementos são retirados.

condicao: Não há condição neste exemplo.

Exposição

Exemplo:

Vamos reescrever o código e comparar:

```python
palavras = ['python', 'é', 'incrível']
maiusculas = [palavra.upper() for palavra in palavras]

print(maiusculas)
```

```
['PYTHON', 'É', 'INCRÍVEL']
```

```python
palavras = ['python', 'é', 'incrível']
maiusculas = []
for palavra in palavras:
    maiusculas.append(palavra.upper())

print(maiusculas)
```

```
['PYTHON', 'É', 'INCRÍVEL']
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Exemplo:

4. Expressões complexas

```
pares_soma = [(x, y, x + y) for x in range(2) for y in range(2, 4) if (x + y) % 2 == 0]
print(pares_soma)
```

```
[(0, 2, 2), (1, 3, 4)]
```

```
[expressao for elemento in lista_original if condicao]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Tome nota:

expressao: (x, y, x + y) – A expressão que cria uma tupla com x, y e a soma deles.

elemento: for x in range(2) for y in range(2, 4) – As variáveis x e y representam os elementos nas listas originais.

lista_original: Não há lista original neste caso; são usadas duas iterações.

condicao: if (x + y) % 2 == 0 – A condição que verifica se a soma de x e y é par.

Exposição

Exemplo:

4. Expressões complexas

```
pares_soma = [(x, y, x + y) for x in range(2) for y in range(2, 4) if (x + y) % 2 == 0]
print(pares_soma)
```

```
[(0, 2, 2), (1, 3, 4)]
```

```
pares_soma = []
for x in range(2):
    for y in range(2, 4):
        if (x + y) % 2 == 0:
            pares_soma.append((x, y, x + y))

print(pares_soma)
```

```
[(0, 2, 2), (1, 3, 4)]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1. O conhecimento sobre a criação de listas de forma eficiente, aplicando o conceito *List Comprehension* (compreensão de listas);

2. A revisão de conteúdo sobre listas;

3. A análise comparativa de vantagens e desvantagens na aplicação de compreensão de listas.

Saiba mais

Você lembra como fazer uma cópia de uma lista em Python?

FELIPE, A. *Python*: Append ou Extend? Alura, 23 jan. 2017. Adicionando elementos na lista. Disponível em: https://www.alura.com.br/artigos/adicionando-elementos-na-lista-do-python-append-ou-extend. Acesso em: 24 mar. 2024.

ORESTES, Y. *Como fazer uma cópia de uma lista no Python*. Alura, 6 set. 2018. Disponível em: https://www.alura.com.br/artigos/como-fazer-copia-de-lista-python. Acesso em: 24 mar. 2024.

Referências da aula

MENEZES, N. N. C. *Introdução à programação com Python*: algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2019.

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Estrutura de controle de fluxo

Compreensão de listas

Aula 2

Código da aula: [DADOS]ANO1C2B2S13A2

Exposição

Objetivos da Aula

Revisar o conceito de dicionário e aprender compreensão de dicionário.

Competências da Unidade (Técnicas e Socioemocionais)

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados.
- Usar técnicas para explorar e analisar dados, aplicar modelos estatísticos, identificar padrões, realizar inferências e tomar decisões fundamentadas em evidências.
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados; trabalhar em equipes multifuncionais, colaborando com colegas, gestores e clientes.

Recursos Didáticos

- Recurso audiovisual para exibição de vídeos e imagens.
- Acesso ao laboratório de informática e/ou à internet.
- Software Anaconda/Jupyter Notebook instalado ou similar.

Duração da Aula

50 minutos

Exposição

Relembrando

	Dicionário
Definição	• Dicionários são estruturas de dados em Python que armazenam pares chave-valor. • Cada valor é associado a uma chave única, proporcionando acesso rápido e eficiente aos dados.
Sintaxe básica	• Declarados usando chaves {}. • Exemplo: meu_dicionario = {'chave1': valor1, 'chave2': valor2}
Acesso aos elementos	• Acesso usando chaves: valor = meu_dicionario['chave']. • As chaves podem ser de diferentes tipos (inteiros, strings etc.).
Operações comuns	• Adição de itens: meu_dicionario[nova_chave] = novo_valor. • Remoção de itens: del meu_dicionario[chave_a_ser_removida]. • Verificação de existência: chave_existe = chave in meu_dicionario.
Mutabilidade	• Dicionários são mutáveis, ou seja, podem ser modificados após a criação.
Uso versátil	• Dicionários são úteis para mapear informações complexas e são amplamente utilizados em Python para representar dados estruturados.

Exposição

Compreensão de dicionários

A compreensão de dicionário em Python é uma construção sintática que permite criar dicionários de maneira concisa e expressiva. Assim como a compreensão de lista, ela simplifica a criação e manipulação de dicionários em uma única linha de código.

Sintaxe básica:

```
{novo_valor: expressao for elemento in lista_original if condicao}
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Tome nota:

novo_valor: O valor associado a uma nova chave no dicionário.
expressao: A expressão que define o valor associado a cada nova chave.
elemento: A variável que representa cada elemento na lista original.
lista_original: A lista da qual os elementos são retirados para criar as chaves no dicionário.
condicao (opcional): Uma condição que determina se um par chave-valor será incluído no dicionário.

Exposição

Exemplo

Neste exemplo, a compreensão de dicionário é usada para criar um dicionário em que cada chave é um número de 1 a 5, e cada valor é o quadrado desse número. A construção é eficiente e legível, tornando-se uma alternativa elegante para loops tradicionais na criação de dicionários.

```
quadrados_dict = {x: x**2 for x in range(1, 6)}
print(quadrados_dict)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Outra forma de criar o mesmo dicionário:

```
quadrados_dict = {}
for x in range(1, 6):
    quadrados_dict[x] = x**2

print(quadrados_dict)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Vamos fazer uma **atividade**

Exercício

Acompanhe as orientações ao lado para realizar esta atividade.

20 minutos

Em grupo

1. Crie uma lista dos **quadrados** dos números de 1 a 10.

2. Gere uma lista com os **cubos** dos números **pares** de 1 a 10.

3. Crie uma lista contendo apenas os **números ímpares** de 1 a 20.

4. Utilizando compreensão de lista, obtenha os primeiros cinco **múltiplos de 3**.

5. Gere uma lista com os números de 1 a 50, **excluindo** os **divisíveis por 5.**

Vamos fazer uma **atividade**

Exercício

Acompanhe as orientações ao lado para realizar esta atividade.

20 minutos

Em grupo

6 Crie um dicionário com **pares** chave-valor, representando o **mapeamento** de números de 1 a 5 para seus **quadrados.**

7 Gere um dicionário em que as **chaves** são os números de 1 a 10, e os valores são seus **cubos.**

8 Crie um dicionário com os números de 1 a 10 como **chaves** e seus **quadrados** como valores, mas apenas para **números ímpares.**

9 Utilizando compreensão de dicionário, **mapeie** os números de 1 a 5 para **"par"** se forem pares, e **"ímpar"** se forem ímpares.

10 Crie um dicionário em que as chaves são os números de 1 a 10, e os valores são **True** se o número for par, e **False** se o número for ímpar.

Registro

Vamos fazer um **quiz**

O que é compreensão de lista em Python?

- Um método para converter listas em *strings*.
- Uma técnica para criar listas de maneira mais concisa e legível.
- Uma função que retorna o comprimento de uma lista.
- Uma maneira de ordenar listas em ordem decrescente.

Registro

Vamos fazer um **quiz**

O que é compreensão de lista em Python?

- ✗ **Um método para converter listas em *strings*.**
- ✓ **Uma técnica para criar listas de maneira mais concisa e legível.**
- ✗ **Uma função que retorna o comprimento de uma lista.**
- ✗ **Uma maneira de ordenar listas em ordem decrescente.**

FEEDBACK GERAL DA ATIVIDADE

Compreensão de lista é uma construção em Python que permite criar listas de forma mais concisa, substituindo a necessidade de usar loops tradicionais.

Registro

Vamos fazer um **quiz**

Como é definida a compreensão de lista no Python?

- [elemento for lista_original]
- {elemento for lista_original}
- [expressao for elemento in lista_original if condicao]
- {expressao for elemento in lista_original if condicao}

Registro

Vamos fazer um **quiz**

Como é definida a compreensão de lista no Python?

- ✗ [elemento for lista_original]
- ✗ {elemento for lista_original}
- ✓ [expressao for elemento in lista_original if condicao]
- ✗ {expressao for elemento in lista_original if condicao}

FEEDBACK GERAL DA ATIVIDADE

A sintaxe básica da compreensão de lista inclui uma expressão que define os elementos da nova lista, um loop que itera sobre uma lista original e uma condição opcional para filtrar os elementos.

Registro

Vamos fazer um **quiz**

Como é a sintaxe básica da compreensão de dicionário em Python?

- {chave, valor for elemento in lista_original}
- [chave: valor for elemento in lista_original]
- {chave: expressao for elemento in lista_original if condicao}
- [chave: valor for elemento in lista_original if condicao]

Registro

Vamos fazer um **quiz**

Como é a sintaxe básica da compreensão de dicionário em Python?

- ✗ {chave, valor for elemento in lista_original}
- ✗ [chave: valor for elemento in lista_original]
- ✓ {chave: expressao for elemento in lista_original if condicao}
- ✗ [chave: valor for elemento in lista_original if condicao]

FEEDBACK GERAL DA ATIVIDADE

A sintaxe básica da compreensão de dicionário inclui uma expressão para a chave e o valor, um loop que itera sobre uma lista original e uma condição opcional para filtrar os elementos.

© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1. O conhecimento sobre o conceito de compreensão de dicionário;

2. A revisão sobre a definição de dicionário;

3. A aplicação prática de compreensão de dicionário por meio de exercícios.

Saiba mais

Quer saber mais sobre dicionário e aprender sobre tupla?

ORESTES, Y. *Python*: trabalhando com dicionários. Alura, 16 out. 2018. Disponível em: https://www.alura.com.br/artigos/trabalhando-com-o-dicionario-no-python. Acesso em: 24 mar. 2024.

Você acha que está programando demais? Entenda neste artigo se um cientista de dados programa.

SILVA, P. H. C. M. da. Cientista de dados programa? Alura, 20 jun. 2023. Disponível em: https://www.alura.com.br/artigos/cientista-de-dados-programa. Acesso em: 24 mar. 2024.

Referências da aula

MENEZES, N. N. C. *Introdução à programação com Python*: algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2019.

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

S13 – Aula 2 – Quiz

Condições de conclusão

Ver

O que é compreensão de lista em Python?

- ○ Uma maneira de ordenar listas em ordem decrescente.
- ○ Um método para converter listas em strings.
- ○ Uma função que retorna o comprimento de uma lista.
- ○ Uma técnica para criar listas de maneira mais concisa e legível.

Como é definida a compreensão de lista no Python?

- ○ [expressao for elemento in lista_original if condicao]
- ○ {expressao for elemento in lista_original if condicao}
- ○ [elemento for lista_original]
- ○ {elemento for lista_original}

Como é a sintaxe básica da compreensão de dicionário em Python?

- ○ [chave: valor for elemento in lista_original if condicao]
- ○ {chave, valor for elemento in lista_original}
- ○ [chave: valor for elemento in lista_original]
- ○ {chave: expressao for elemento in lista_original if condicao}

Disciplina
Programação Aplicada a Ciência de Dados 2º Bimestre

Curso
Técnico em Ciência de Dados

Ano letivo
2025

Retornar ao Sumário

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Estrutura de controle de fluxo

Compreensão de listas

Aula 3

Código da aula: [DADOS]ANO1C2B2S13A3

Exposição

Objetivos da Aula

Praticar o conhecimento de compreensão de listas e dicionários.

Competências da Unidade (Técnicas e Socioemocionais)

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados.
- Usar técnicas para explorar e analisar dados, aplicar modelos estatísticos, identificar padrões, realizar inferências e tomar decisões fundamentadas em evidências.
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados; trabalhar em equipes multifuncionais, colaborando com colegas, gestores e clientes.

Recursos Didáticos

- Recurso audiovisual para exibição de vídeos e imagens.
- Acesso ao laboratório de informática e/ou à internet.
- Software Anaconda/Jupyter Notebook instalado ou similar.

Duração da Aula

50 minutos

Registro

Vamos fazer uma **atividade**

Elaborado especialmente para o curso com a ferramenta Microsoft Copilot.

Transação financeira

Você trabalha em uma empresa financeira e observou o código de um programador que criou um programa em Python para analisar as transações financeiras. Sabe-se que transações positivas são depósitos (valores positivos) e valores negativos são retiradas de dinheiro.

Seu gestor pediu para você reescrever o código e **NÃO** usar compreensão de lista. **Como o código ficaria?**

Envie o código no AVA.

Vamos fazer uma **atividade**

Exercício

Acompanhe as orientações ao lado para realizar esta atividade.

35 minutos

Em grupo

1. Gere uma lista dos quadrados dos números de **1 a 10** e armazene em um **dicionário** em que as **chaves** são os **números,** e os **valores** são seus **quadrados. Use compreensão de lista**.

2. Crie uma **lista** com os números de **1 a 20.** Transforme-a em um **dicionário** em que as **chaves** são os **números,** e os **valores** são **"par" ou "ímpar". Use compreensão de lista.**

3. Utilizando compreensão de lista, crie uma lista com o **dobro** dos números de 1 a 10, mas apenas para **números ímpares**.

4. Gere um **dicionário** em que as chaves são os números de 1 a 10, e os valores são **True** se o número for **primo** e **False** se **não for.** Use **compreensão de dicionário.**

Vamos fazer uma **atividade**

Exercício

Acompanhe as orientações ao lado para realizar esta atividade.

35 minutos

Em grupo

5 Crie uma lista contendo os números de 1 a 20, mas **substitua os múltiplos de 3 por "Fizz"** e os **múltiplos de 5 por "Buzz"**. Use compreensão de lista.

6 **Reescreva** os exercícios de 1 a 5 **sem usar** compreensão de lista.

7 Crie um **dicionário** com as letras de uma **string** como **chaves,** e o **número de vezes que cada letra aparece** como **valores.**

8 Utilizando **compreensão de lista,** crie uma lista de todas as palavras **com mais de cinco letras em uma frase.**

9 Gere um **dicionário** em que as chaves são os **meses do ano,** e os valores são o **número de dias em cada mês.**

© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1. As aplicações dos conceitos de compreensão de lista e Dicionário;

2. A prática de vários exercícios que envolvia o conceito de compreensão.

Saiba mais

Código limpo, será que você consegue deixar seu código assim?

FURTADO, F. *Clean code*: O que é, casos de uso, exemplo de código limpo. Alura, 24 abr. 2019. Disponível em: https://www.alura.com.br/artigos/o-que-e-clean-code. Acesso em: 24 mar. 2024.

Referências da aula

MENEZES, N. N. C. *Introdução à programação com Python*: algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2019.

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

S13 – Aula 3 – Registro

Transação financeira

Você trabalha em uma empresa financeira e observou o código de um programador que criou um programa em Python para analisar as transações financeiras. Sabe-se que transações positivas são depósitos (valores positivos) e valores negativos são retiradas de dinheiro.

Seu gestor pediu para você reescrever o código e NÃO usar compreensão de lista. Como o código ficaria?

Envie o código no AVA.

Condições de conclusão

Fazer um envio

Resumo das Avaliações

Turmas separadas: 293566972 | 2ª SERIE BT MANHA ANUAL | 99 | JOAO CRUZ PROF

Oculto para estudantes	Não
Participantes	43
Enviado	0
Precisa ser avaliado	0

Disciplina
Programação Aplicada a Ciência de Dados 2º Bimestre

Curso
Técnico em Ciência de Dados

Ano letivo
2025

Retornar ao Sumário

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Estrutura de controle de fluxo

Compreensão de listas

Aula 4

Código da aula: [DADOS]ANO1C2B2S13A4

Exposição

Objetivos da Aula

Conhecer as boas práticas do Python e do código limpo.

Competências da Unidade (Técnicas e Socioemocionais)

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados.
- Usar técnicas para explorar e analisar dados, aplicar modelos estatísticos, identificar padrões, realizar inferências e tomar decisões fundamentadas em evidências.
- Colaborar efetivamente com outros profissionais, como cientistas de dados e engenheiros de dados; trabalhar em equipes multifuncionais colaborando com colegas, gestores e clientes.

Recursos Didáticos

- Recurso audiovisual para exibição de vídeos e imagens.
- Acesso ao laboratório de informática e/ou à internet.
- Software Anaconda/Jupyter Notebook instalado ou similar.

Duração da Aula

50 minutos

Motivação: código limpo

O seu colega de trabalho saiu de férias e você precisa usar o código dele. O código a ser utilizado encontra-se a seguir.

O que você vai fazer? Ligar para o colega? Falar com seu gestor? Desistir?

```
def fatorial(n):if n==0 or n==1:return 1
else:return n*fatorial(n-1)
numeros=[]
for i in range(5):numeros.append(i+1)
print(numeros)
if numeros[0]<3:print("menor que 3")
else:print("maior ou igual a 3")
fatoriais = []
for i in numeros:fatoriais.append(fatorial(i))
print(fatoriais)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Momento de **reflexão**

Exposição

Código limpo

Código limpo refere-se:

- a um estilo de programação que promove a **legibilidade**, a **simplicidade** e a **manutenibilidade** do código-fonte.
- àquele que é fácil de entender, modificar e manter. Ele segue boas práticas e princípios de design de software que visam criar um software eficiente, robusto e de alta qualidade.

Tome nota:

O código limpo não é apenas uma questão de estética; é uma prática que contribui para a eficiência no desenvolvimento de software e para a criação de sistemas mais robustos e sustentáveis ao longo do tempo.

Exposição

Código limpo

Alguns dos princípios e características associados ao código limpo incluem:

	Princípios e características
Nomes significativos	Escolher nomes de variáveis, funções, classes e métodos que sejam descritivos e que reflitam o propósito e a função do elemento.
Funções e métodos concisos	Manter funções e métodos curtos e com foco em uma única responsabilidade. Evitar funções muito extensas que realizam diversas tarefas.
Comentários úteis	Utilizar comentários, quando necessário, para explicar partes complexas do código ou fornecer informações adicionais. Comentários devem ser claros e relevantes.
Indentação consistente	Adotar uma indentação consistente para melhorar a legibilidade do código. Isso facilita a visualização da estrutura e do fluxo do programa.
Evitar códigos repetitivos	Evitar duplicação de código sempre que possível. Utilizar abstrações como funções, classes e módulos para promover a reutilização.

Exposição

Código limpo

Alguns dos princípios e características associados ao código limpo incluem:

	Princípios e características
Testabilidade	Escrever um código que seja fácil de testar. Códigos testáveis tendem a ser mais robustos e menos propensos a erros.
Organização lógica	Organizar o código de maneira lógica, agrupando funcionalidades relacionadas e mantendo uma estrutura coerente.
Refatoração	Estar disposto a refatorar o código, conforme necessário, para melhorar sua estrutura e clareza, sem alterar seu comportamento.
Documentação adequada	Incluir documentação, como docstrings, em funções e módulos, para fornecer informações sobre como usar o código.
Atenção aos detalhes de estilo	Seguir as convenções de estilo da linguagem de programação utilizada. Isso inclui a formatação adequada do código, o uso consistente de espaços em branco e outros detalhes estilísticos.

Exposição

PEP 8

O PEP 8 (Python Enhancement Proposal 8) é um guia de estilo para a escrita de código em Python. Ele fornece:

- ✓ recomendações sobre a formatação do código.
- ✓ a escolha de nomes de variáveis.
- ✓ a estruturação de imports.
- ✓ a organização de código.
- ✓ outras convenções que visam melhorar a legibilidade e a consistência do código Python.

Dica:

O PEP 8 é amplamente aceito na comunidade Python e é seguido para manter a consistência no ecossistema Python. Ferramentas podem ajudar a automatizar a verificação do código em relação às diretrizes do PEP 8. Adotar essas convenções ajuda a melhorar a colaboração e a legibilidade do código em projetos Python.

Exposição

PEP 8

Algumas das principais diretrizes do PEP 8 incluem:

1

Indentação

Use espaços em branco para a indentação em vez de *tabs*. A convenção recomendada é usar quatro espaços para cada nível de indentação.

2

Comprimento das linhas

Limite o comprimento das linhas a 79 caracteres para código, e 72 para docstrings. Se uma expressão não couber em uma única linha, use parênteses para dividir em várias linhas.

3

Importações

Agrupe as importações em três seções, sendo elas: bibliotecas padrão, bibliotecas relacionadas a terceiros e suas próprias bibliotecas.

4

Espaços em branco em expressões e instruções

Use espaços em branco de forma consistente para melhorar a legibilidade. Evite espaços em branco em excesso.

Exposição

PEP 8

Algumas das principais diretrizes do PEP 8 incluem:

5

Nomeação de variáveis e funções

Use nomes descritivos e evite nomes únicos que possam ser confundidos com palavras-chave do Python.

6

Comentários

Use comentários para explicar partes complexas do código, mas evite comentários óbvios e redundantes.

7

Docstrings

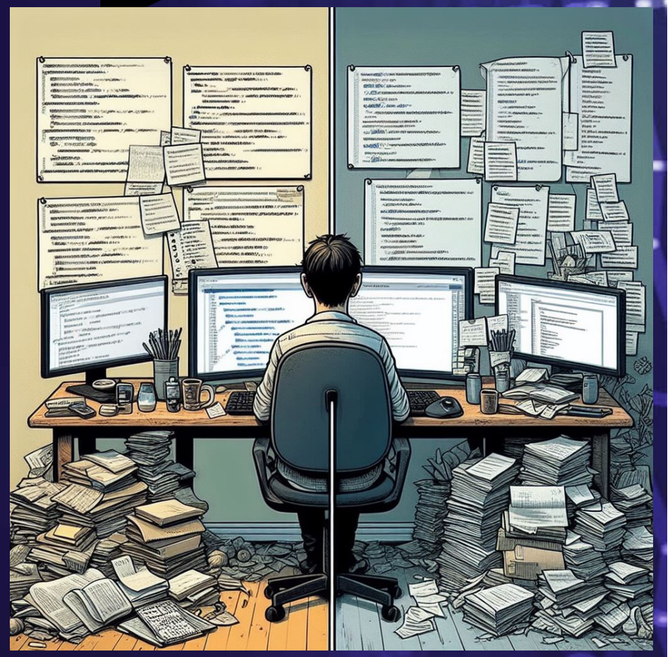
Inclua docstrings para documentar classes, funções e módulos. Siga o formato do Docstring Convention PEP 257.

8

Operadores

Adicione espaços em torno de operadores para melhorar a clareza.

Vamos fazer uma **atividade**



Elaborado especialmente para o curso com a ferramenta Microsoft Copilot.

Secretaria da Educação SÃO PAULO GOVERNO DO ESTADO

Código limpo

O seu colega de trabalho saiu de férias e você precisa usar o código dele. O código do seu colega encontra-se a seguir:

```
def fatorial(n):if n==0 or n==1:return 1
else:return n*fatorial(n-1)
numeros=[]
for i in range(5):numeros.append(i+1)
print(numeros)
if numeros[0]<3:print("menor que 3")
else:print("maior ou igual a 3")
fatoriais = []
for i in numeros:fatoriais.append(fatorial(i))
print(fatoriais)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

- **Reescreva** o código, deixando-o limpo e legível. Use compreensão de lista, se couber.

© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1. O conhecimento sobre as boas práticas em Python;

2. A aplicação do guia de estilo para programar em Python (PEP8);

3. A compreensão do conceito de código limpo.

Saiba mais

Onde aplicar a compreensão de listas?

FELIPE, A. *Compreensão de listas no Python*. Alura, 25 jul. 2016. Disponível em: https://www.alura.com.br/artigos/simplicando-o-processamento-com-compreensao-de-lista-do-python. Acesso em: 24 mar. 2024.

Um artigo completo sobre listas:

ORESTES, Y. *Listas em Python*: operações básicas. Alura, 18 set. 2023. Disponível em: https://www.alura.com.br/artigos/listas-no-python#filtrando-uma-lista-utilizando-_list-comprehensions_. Acesso em: 24 mar. 2024.

Boas práticas em Python:

VAN ROSSUM, G.; WARSAW, B.; COGHLAN, A. *PEP 8 – Style Guide for Python Code*. Python Enhancement Proposals, 5 jul. 2001. Disponível em: https://peps.python.org/pep-0008/. Acesso em: 24 mar. 2024.

Referências da aula

BRANCO, H, J. *PEP 8 e imports em Python*. Medium, 9 jul. 2021. Disponível em: https://medium.com/gbtech/pep-8-e-imports-em-python-78a6fbf53475. Acesso em: 24 mar. 2024.

MARTIN, R. C. *Código limpo*: habilidades práticas do Agile Software. Rio de Janeiro: Alta Books, 2009.

VAN ROSSUM, G.; WARSAW, B.; COGHLAN, A. *PEP 8 – Style Guide for Python Code*. Python Enhancement Proposals, 5 jul. 2001. Disponível em: https://peps.python.org/pep-0008/. Acesso em: 24 mar. 2024.

VAN ROSSUM, G.; WARSAW, B.; COGHLAN, A. *PEP 8 - Guia de estilo para Python*. Traduzido por Pedro Werneck, 16 nov. 2003. Disponível em: https://wiki.python.org.br/GuiaDeEstilo. Acesso em: 24 mar. 2024.

Identidade visual: imagens © Getty Images.

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO