

E d u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados

Estrutura de controle de fluxo

Revisão e exercícios práticos

Aula 1

Código da aula: [DADOS]ANO1C2B2S15A1



Objetivo da aula

Revisar o conteúdo de condicionais, loops, funções, listas, dicionários, tuplas e conjuntos em Python.



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 com base 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

Motivação

Agora que sabemos muitos conceitos, podemos juntá-los e criar programas que resolvem diversos **problemas reais**. Veja abaixo um exemplo de um gerenciador de atividades.

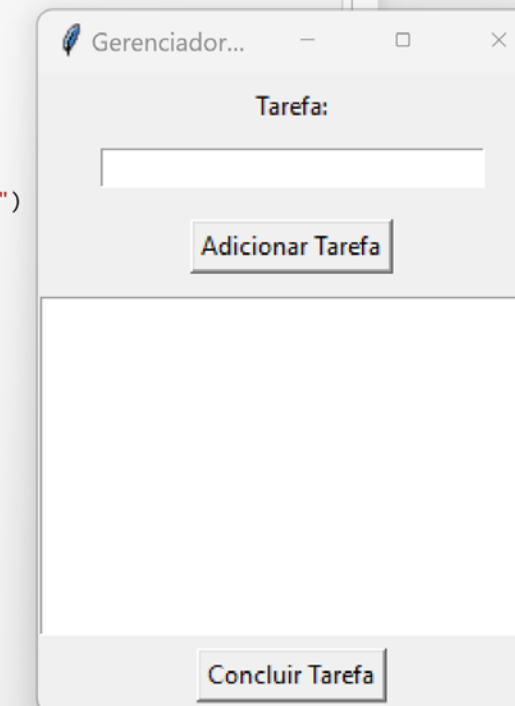
```
: ▶ import tkinter as tk
from tkinter import messagebox
from datetime import datetime

tarefas = []

def adicionar_tarefa():
    tarefa = entry_tarefa.get()
    if tarefa:
        tarefas.append({"tarefa": tarefa, "data": datetime.now()})
        listbox_tarefas.insert(tk.END, f"{tarefa} - {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
        entry_tarefa.delete(0, tk.END)
    else:
        messagebox.showwarning("Aviso", "Digite uma tarefa.")

def concluir_tarefa():
    indice_selecionado = listbox_tarefas.curselection()
    if indice_selecionado:
        tarefa_concluida = tarefas.pop(indice_selecionado[0])
        listbox_tarefas.delete(indice_selecionado)
        messagebox.showinfo("Concluído", f"Tarefa '{tarefa_concluida['tarefa']}' concluída!")

# Interface gráfica
root = tk.Tk()
root.title("Gerenciador de Tarefas")
label = tk.Label(root, text="Tarefa:")
label.pack(pady=5)
entry_tarefa = tk.Entry(root, width=30)
entry_tarefa.pack(pady=5)
botao_adicionar = tk.Button(root, text="Adicionar Tarefa", command=adicionar_tarefa)
botao_adicionar.pack(pady=10)
listbox_tarefas = tk.Listbox(root, width=40, height=10)
listbox_tarefas.pack()
botao_concluir = tk.Button(root, text="Concluir Tarefa", command=concluir_tarefa)
botao_concluir.pack(pady=5)
root.mainloop()
```



Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição



© Getty Images

Revisão

Inicialmente, abordaremos as **estruturas condicionais**, que permitem que seu código faça escolhas com base em condições lógicas.

Em seguida, mergulharemos nos **loops**, mecanismos fundamentais para repetir blocos de código.

Discutiremos também a **criação e a utilização de funções**, peças-chave na modularização e na organização do código.

Após isso, entraremos nas estruturas de dados, começando com **listas versáteis, dicionários associativos, tuplas imutáveis e conjuntos únicos**.

A função **enumerate** ajudará a iterar sobre sequências enquanto capturamos índices e valores.

Por fim, exploraremos a poderosa compreensão de listas, uma forma concisa de criar listas em Python.

Revisão – Estruturas condicionais

Lembrando:

As estruturas condicionais são utilizadas para **controlar** o **fluxo do programa** com base em condições lógicas.

Operadores de comparação:

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

```
if condição:  
    # código a ser executado se a condição for verdadeira  
elif outra_condição:  
    # código a ser executado se a segunda condição for verdadeira  
else:  
    # código a ser executado se nenhuma das condições anteriores for verdadeira
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exemplo – Estruturas condicionais

Escreva um programa que avalie a idade do usuário e imprima se ele é menor de idade, tem 18 anos ou é maior de idade.

```
idade = 18

if idade < 18:
    print("Você é menor de idade.")
elif idade == 18:
    print("Você tem 18 anos.")
else:
    print("Você é maior de idade.")
```

Você tem 18 anos.

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Loops

Os loops permitem a **repetição** de **blocos de código**.

while condição:

```
# código a ser repetido enquanto a condição for verdadeira  
# certifique-se de ter uma lógica para alterar a condição
```

for elemento **in** sequência:

```
# código a ser executado para cada elemento na sequência
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Loops

Crie um programa que itere sobre uma lista de frutas e imprima cada uma delas.

```
frutas = ["maçã", "banana", "laranja"]  
  
for fruta in frutas:  
    print(fruta)
```

```
maçã  
banana  
laranja
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Loops

As funções ajudam a **organizar** e a **reutilizar código**.

```
def nome_da_funcao(parâmetros):  
    # código da função  
    return resultado
```

```
resultado = nome_da_funcao(argumentos)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exemplo – Funções

Defina uma função que receba o nome de uma pessoa como argumento e retorne uma mensagem de saudação personalizada.

```
def saudacao(nome):  
    return f"Olá, {nome}!"  
  
mensagem = saudacao("João")  
print(mensagem)
```

Olá, João!

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Listas

Listas são **estruturas** de dados versáteis que podem **armazenar múltiplos elementos**.

```
lista = [elemento1, elemento2, elemento3]
primeiro_elemento = lista[0]
lista[1] = novo_valor
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exemplo – Listas

Crie uma lista de números e demonstre como acessar e modificar elementos da lista.

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

terceiro_numero = numeros[2]
print(terceiro_numero)

numeros[1] = 10
print(numeros)
```

```
3
[1, 10, 3, 4, 5]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Compreensão de listas

Listas são estruturas de dados versáteis que podem armazenar **múltiplos elementos**.

```
nova_lista = [expressao for elemento in sequência if condição]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Compreensão de listas

Crie uma lista de números e, utilizando a compreensão de lista, gere uma nova lista contendo o quadrado de cada número.

```
numeros = [1, 2, 3, 4, 5]  
  
quadrados = [x**2 for x in numeros]  
print(quadrados)
```

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

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Tuplas

Tuplas são **sequências imutáveis** de elementos.

```
tupla = (elemento1, elemento2, elemento3)  
primeiro_elemento = tupla[0]
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Tuplas

Declare uma tupla representando as coordenadas de um ponto e mostre como acessar seus elementos.

```
coordenadas = (3, 4)

x, y = coordenadas
print(f"Coordenadas: x={x}, y={y}")
```

Coordenadas: x=3, y=4

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Enumerate

A função **enumerate** é útil para obter **tanto o índice quanto o valor** durante a iteração.

```
for indice, elemento in enumerate(sequência):  
    # código a ser executado para cada elemento na sequência
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exemplo – Enumerate

Utilize a função `enumerate` para percorrer uma lista de frutas, exibindo o índice e o nome de cada fruta.

```
frutas = ["maçã", "banana", "laranja"]  
  
for indice, fruta in enumerate(frutas):  
    print(f"Índice: {indice}, Fruta: {fruta}")
```

```
Índice: 0, Fruta: maçã  
Índice: 1, Fruta: banana  
Índice: 2, Fruta: laranja
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Conjuntos

Conjuntos são **coleções** de **elementos únicos**.

```
conjunto = {elemento1, elemento2, elemento3}  
conjunto.add(novo_elemento)  
conjunto.remove(elemento)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exemplo – Conjuntos

Crie um conjunto de cores primárias e demonstre como adicionar e remover elementos.

```
cores_primarias = {'vermelho', 'azul', 'amarelo'}  
  
cores_primarias.add('verde')  
print(cores_primarias)  
  
cores_primarias.remove('azul')  
print(cores_primarias)  
  
{'azul', 'verde', 'amarelo', 'vermelho'}  
{'verde', 'amarelo', 'vermelho'}
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Dicionários

Dicionários são estruturas que **associam chaves** a **valores**.

```
dicionario = {'chave1': valor1, 'chave2': valor2}  
valor = dicionario['chave']  
dicionario['chave'] = novo_valor
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Revisão – Dicionários

Defina um dicionário representando informações sobre uma pessoa e mostre como acessar e modificar valores associados às chaves.

```
pessoa = {'nome': 'Ana', 'idade': 25, 'cidade': 'São Paulo'}

idade_da_pessoa = pessoa['idade']
print(idade_da_pessoa)

pessoa['cidade'] = 'Rio de Janeiro'
print(pessoa)
```

25

```
{'nome': 'Ana', 'idade': 25, 'cidade': 'Rio de Janeiro'}
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.



Vamos
fazer um
quiz

**Qual é a estrutura básica de
uma condicional em Python?**

if, else, elif

start, middle, end

check, pass, fail

case, when, otherwise



Vamos
fazer um
quiz

Qual é a estrutura básica de uma condicional em Python?



if, else, elif

start, middle, end



check, pass, fail

case, when, otherwise



FEEDBACK GERAL DA ATIVIDADE

if, else, elif – Esta é a estrutura básica de uma condicional em Python, permitindo a execução condicional de diferentes blocos de código com base em condições.



Vamos
fazer um
quiz

Qual é a principal finalidade de uma função em Python?

Repetir blocos de código.

Modificar valores em uma lista.

Organizar e reutilizar código.

Adicionar elementos a um conjunto.



Vamos
fazer um
quiz

Qual é a principal finalidade de uma função em Python?



Repetir blocos de código.

Modificar valores em uma lista.



Organizar e reutilizar código.

Adicionar elementos a um conjunto.



FEEDBACK GERAL DA ATIVIDADE

Em Python, a função é usada principalmente para organizar e reutilizar um código, facilitando a manutenção e melhorando a legibilidade.



Vamos
fazer um
quiz

Para que é utilizada a função `enumerate` em Python?

Adicionar elementos
a uma lista.

Contar o número de elementos
em uma lista.

Gerar uma numeração de
pares índice-valor durante a
iteração.

Eliminar elementos duplicados
de uma lista.



Vamos
fazer um
quiz

Para que é utilizada a função `enumerate` em Python?



Adicionar elementos
a uma lista.

Contar o número de elementos
em uma lista.



Gerar uma numeração de
pares índice-valor durante a
iteração.

Eliminar elementos duplicados
de uma lista.



FEEDBACK GERAL DA ATIVIDADE

A função `enumerate` é usada para obter tanto o índice quanto o valor durante a iteração de uma sequência.



© Getty Images

O que nós
**aprendemos
hoje?**

Hoje desenvolvemos

- 1** Revisão dos conceitos de programação em Python;
- 2** Compreensão dos fundamentos de Python abordados nas aulas anteriores.

Saiba mais

Quer saber como praticar mais de programação? Ouça esse podcast, em que Lívia, uma menina de 16 anos, conta como já programou uma quantidade razoável de Python e C++.

SCUBA PONTO DEV. A jovem desenvolvedora: Lívia Scopel, ep. 16. *Alura*, 25 fev. 2021. Disponível em: <https://cursos.alura.com.br/extra/scubadev/a-jovem-desenvolvedora-livia-scopel-scuba-ponto-dev-16-a798>. Acesso em: 5 abr. 2024.

Referências da aula

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

Identidade visual: Imagens © Getty Images

Ed u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados



S15 – Aula 1 – Quiz

Condições de conclusão

Ver

Qual é a estrutura básica de uma condicional em Python?

- ☐ if, else, elif
- ☐ start, middle, end
- ☐ case, when, otherwise
- ☐ check, pass, fail

Qual é a principal finalidade de uma função em Python?

- ☐ Organizar e reutilizar código.
- ☐ Modificar valores em uma lista.
- ☐ Repetir blocos de código.
- ☐ Adicionar elementos a um conjunto.

Para que é utilizada a função enumerate em Python?

- ☐ Adicionar elementos a uma lista.
- ☐ Contar o número de elementos em uma lista.
- ☐ Eliminar elementos duplicados de uma lista.
- ☐ Gerar uma numeração de pares índice-valor durante a iteração.



Disciplina

Programação Aplicada a Ciência de Dados 2º Bimestre

Curso

Técnico em Ciência de Dados

Ano letivo

2025



Retornar ao Sumário



Ed u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados

Estrutura de controle de fluxo

Revisão e exercícios práticos

Aula 2

[DADOS]ANO1C2B2S15A2



Objetivo da aula

- Aplicar os conhecimentos em um projeto.



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 com base 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.

Vamos
fazer uma
atividade

🕒 45 minutos



Elaborado especialmente para o curso
com a ferramenta Microsoft Copilot.

Projeto: Agenda de contatos – Parte 1

- 1 Reúnam-se em **grupos de quatro pessoas** e façam as atividades.
- 2 Utilizem o **Roteiro de Atividade Prática** e continuem trabalhando na construção solicitada.
- 3 Essa atividade **continuará nas próximas aulas**, então não é necessário fazer a entrega neste momento.

Vamos fazer uma **atividade**

Desenvolva um programa em Python para gerenciar uma agenda de contatos, que deve oferecer as funcionalidades listadas ao lado:

 **45 minutos**

 **Em grupo**

Projeto: Agenda de contatos – Parte 1

- 1 Adicionar contato:** permitir que o usuário adicione novos contatos, fornecendo nome, idade, telefone, e-mail, renda e estado.
- 2 Exibir contatos:** mostrar uma lista completa de todos os contatos na agenda, incluindo nome, telefone e e-mail.
- 3 Buscar contato:** permitir que o usuário busque as informações de um contato específico, fornecendo o nome do contato.
- 4 Remover contato:** possibilitar a remoção de um contato da agenda com base no nome.
- 5 Corrigir contato:** substituir uma informação de um contato.
- 6 Mostrar a quantidade de contatos cadastrados.**
- 7 Mostrar a média de idade dos contatos cadastrados.**
- 8 Mostrar a quantidade de contatos cadastrados por estado.**

Observação: o programa deve permitir que o usuário preencha os contatos até digitar “sair”.

Hoje desenvolvemos

- 1 Aplicação prática dos conceitos que já estudamos, como estruturas condicionais, loops, funções, listas, enumerate, compreensão de lista, tuplas, conjuntos e dicionários.

O que nós
**aprendemos
hoje?**

© Getty Images

Saiba mais

Será que é fácil aprender a programar e seguir em uma carreira dentro da área de tecnologia?

Ouçá este **podcast** e descubra!

HIPSTERS: Começo de carreira, ep. 128. *Alura*, 25 dez. 2018. Disponível em:

<https://www.alura.com.br/podcast/comeco-de-carreira-hipsters-128-a454>. Acesso em: 5 abr. 2024.

Referências da aula

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

Identidade visual: Imagens © Getty Images

Ed u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados

Ed u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados

Estrutura de controle de fluxo

Revisão e exercícios práticos

Aula 3

Código da aula: [DADOS]ANO1C2B2S15A3



Objetivo da aula

- Aplicar o conceito de tuplas em Python.



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 com base 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.



Vamos fazer uma **atividade**

Continue a desenvolver com seu grupo um programa em Python para gerenciar uma agenda de contatos, que deve oferecer as funcionalidades listadas ao lado:

 **45 minutos**

 **Em grupo**

Projeto: Agenda de contatos – Parte 2

- 1 Adicionar contato:** permitir que o usuário adicione novos contatos, fornecendo nome, idade, telefone, e-mail, renda e estado.
- 2 Exibir contatos:** mostrar uma lista completa de todos os contatos na agenda, incluindo nome, telefone e e-mail.
- 3 Buscar contato:** permitir que o usuário busque as informações de um contato específico, fornecendo o nome do contato.
- 4 Remover contato:** possibilitar a remoção de um contato da agenda com base no nome.
- 5 Corrigir contato:** substituir uma informação de um contato.
- 6 Mostrar a quantidade de contatos cadastrados.**
- 7 Mostrar a média de idade dos contatos cadastrados.**
- 8 Mostrar a quantidade de contatos cadastrados por estado.**

Lembre-se de que o programa deve permitir que o usuário preencha os contatos até digitar “sair”.

Hoje desenvolvemos

- 1 Aplicação prática dos conceitos que já estudamos, como estruturas condicionais, loops, funções, listas, enumerate, compreensão de lista, tuplas, conjuntos e dicionários.

O que nós
**aprendemos
hoje?**

© Getty Images

Saiba mais

Por que a linguagem Python está na moda mesmo sem ser nova? Qual é a relação com Machine Learning e Big Data? E na web? E a comunidade? Descubra mais no link abaixo:

HIPSTERS: Python, ep. 122. *Alura*, 13 nov. 2018.

Disponível em:

<https://cursos.alura.com.br/extra/hipsterstech/python-hipsters-122-a460>. Acesso em: 5 abr. 2024.

Referências da aula

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

Identidade visual: Imagens © Getty Images

E d u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados

Ed u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados

Estrutura de controle de fluxo

Revisão e exercícios práticos

Aula 4

[DADOS]ANO1C2B2S15A4



Objetivo da aula

- Aplicar os conceitos de conjuntos (sets) em Python.



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 com base 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.

Vamos
fazer uma
atividade

Com seu grupo, finalize o desenvolvimento de um programa em Python para gerenciar uma agenda de contatos, que deve oferecer as funcionalidades indicadas ao lado:

Projeto: Agenda de contatos – Final

- 1 Adicionar contato:** permitir que o usuário adicione novos contatos, fornecendo nome, idade, telefone, e-mail, renda e estado.
- 2 Exibir contatos:** mostrar uma lista completa de todos os contatos na agenda, incluindo nome, telefone e e-mail.
- 3 Buscar contato:** permitir que o usuário busque as informações de um contato específico, fornecendo o nome do contato.
- 4 Remover contato:** possibilitar a remoção de um contato da agenda com base no nome.
- 5 Corrigir contato:** substituir uma informação de um contato.
- 6 Mostrar a quantidade de contatos cadastrados.**
- 7 Mostrar a média de idade dos contatos cadastrados.**
- 8 Mostrar a quantidade de contatos cadastrados por estado.**

Observação: o programa deve permitir que o usuário preencha os contatos até digitar “sair”.

Hoje desenvolvemos

- 1 Aplicação prática dos conceitos que já estudamos, como estruturas condicionais, loops, funções, listas, enumerate, compreensão de lista, tuplas, conjuntos e dicionários.

O que nós
**aprendemos
hoje?**

© Getty Images

Saiba mais

Quer aprender mais sobre o **ecossistema Python**? Acesse o link abaixo:

HIPSTERS PONTO TEC: Ecossistema Python, ep. 387. *Alura*, 5 dez. 2023. Disponível em: <https://cursos.alura.com.br/extra/hipsterstech/ecossistema-python-hipsters-ponto-tech-387-a9175>. Acesso em: 5 abr. 2024.

Referências da aula

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

Identidade visual: Imagens © Getty Images

Ed u c a ç ã o
P r o f i s s i o n a l
P a u l i s t a

Técnico em
Ciência de
Dados



S15 – Aula 4 – Registro

Projeto: Agenda de contatos – Final

Com seu grupo, finalize o desenvolvimento de um programa em Python para gerenciar uma agenda de contatos, que deve oferecer as funcionalidades indicadas a seguir:

1. **Adicionar contato:** permitir que o usuário adicione novos contatos, fornecendo nome, idade, telefone, e-mail, renda e estado.
2. **Exibir contatos:** mostrar uma lista completa de todos os contatos na agenda, incluindo nome, telefone e e-mail.
3. **Buscar contato:** permitir que o usuário busque as informações de um contato específico, fornecendo o nome do contato.
4. **Remover contato:** possibilitar a remoção de um contato da agenda com base no nome.
5. **Corrigir contato:** substituir uma informação de um contato.
6. **Mostrar a quantidade de contatos cadastrados.**
7. **Mostrar a média de idade dos contatos cadastrados.**
8. **Mostrar a quantidade de contatos cadastrados por estado.**

Observação: o programa deve permitir que o usuário preencha os contatos até digitar “sair”.

Ao finalizar a atividade, envie pelo Ambiente Virtual de Aprendizagem (AVA) o arquivo com extensão .ipynb.

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

