Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Programação aplicada à Ciência de Dados

Estrutura de controle de fluxo

Aula 1

Código da aula: [DADOS]ANO1C2B2S10A1

Exposição

Objetivo da aula

- Relembrar a estrutura de controle de loop.

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 baseadas em evidências.

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: par ou ímpar?

Formem duplas e joguem par ou ímpar.

Caso alguém não saiba como jogar, como podemos ensinar as regras dessa brincadeira?

Getty Images

Exposição

Instruções do Jogo "Par ou Ímpar":

1. **Participantes:**
   Este jogo requer **dois** jogadores: um será designado como o "Jogador 1", e o outro como o "Jogador 2".
2. **Rodada inicial:**
   O Jogador 1 faz a escolha entre **"Par"** ou **"Ímpar"**. Em seguida, ao Jogador 2 atribui-se automaticamente a outra opção.
3. **Números escolhidos:**
Cada jogador escolhe um número inteiro, geralmente de **1 a 10**.
4. **Soma dos números:**
   Os números escolhidos pelos dois jogadores são **somados**.
5. **Determinar Par ou Ímpar:**
O Jogador 1 (aquele que escolheu "Par") ganha se a soma for par. O Jogador 2 (aquele que escolheu "Ímpar") ganha se a soma for ímpar.
6. **Anunciar o vencedor:**
   O jogador, cuja escolha (Par ou Ímpar) corresponde à **paridade** da soma, é declarado o vencedor.
7. **Jogar novamente (opcional):**
   Os jogadores podem optar por jogar várias rodadas, alternando as escolhas entre "Par" e "Ímpar".

Exposição

Controle de loop em Python

Os **loops** permitem a execução repetida de um bloco de código até que uma condição seja atendida. Em Python, existem diferentes estruturas de controle de loop que oferecem flexibilidade e eficiência na implementação de iterações.

Nesta aula, vamos relembrar os principais tipos de loops em Python e explorar como controlar o fluxo de execução dentro deles.

Exposição

Estruturas básicas de loop

O loop **for** é utilizado para iterar sobre uma sequência (como uma lista, tupla, *string* ou *range*). Sua sintaxe básica é:

```
for elemento in sequencia:
    # Código a ser repetido
```

Exemplo:

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

```
maçã
banana
laranja
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Estruturas básicas de loop

O loop **while** executa um bloco de código, enquanto uma condição específica for verdadeira. A estrutura básica é:

```
while condição:
    # Código a ser repetido
    # enquanto condição é verdadeira
```

Exemplo:

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

```
0
1
2
3
4
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Vamos fazer uma **atividade**

Registro

Programar o jogo Par ou Ímpar

Seguindo as instruções da brincadeira Par ou Ímpar, crie em grupo um programa que simule a partida entre você e o computador.

Obs.: para simular a jogada do computador, você pode usar o código abaixo.

```
1  import random
2  random.randint(1, 10)
3
```

```
3
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Dica

A biblioteca **random** é útil para gerar números aleatórios. Exemplo: o comando random.randint(1,10) escolhe um número aleatório entre 1 e 10.

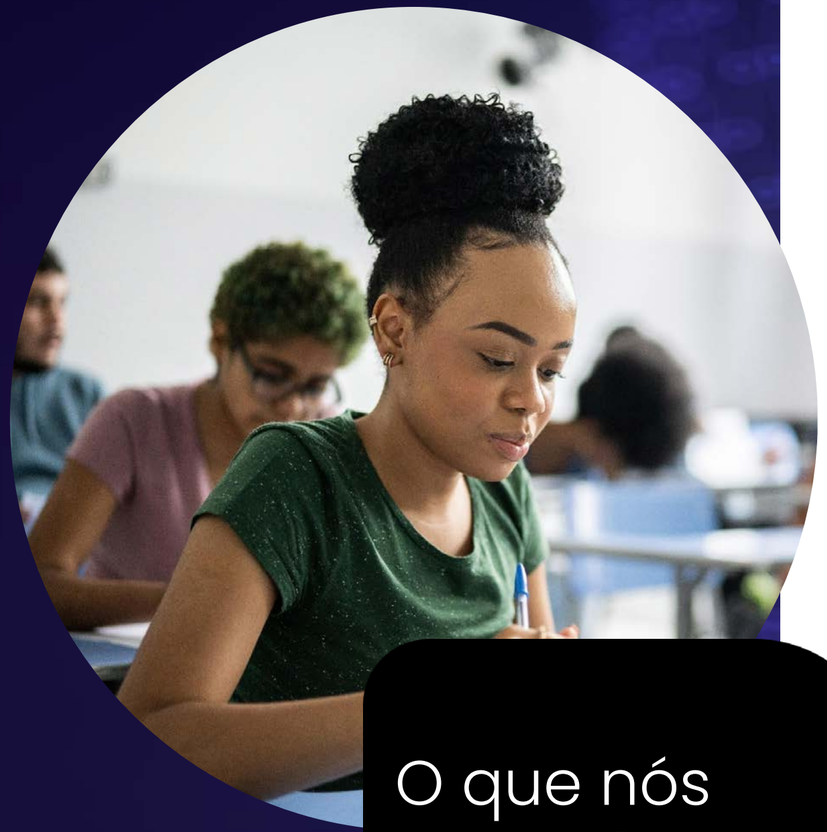
Vamos fazer uma **atividade**

Registro

Programar o jogo Par ou Ímpar

Veja abaixo uma sugestão para a saída do programa:

```
Bem-vindo ao Jogo Par ou Ímpar!

Escolha [P]ar ou [Í]mpar: P
Escolha um número entre 1 e 10: 3

Você escolheu 3 e o computador escolheu 3.
Você ganhou! A soma é 6

Deseja jogar novamente? (s/n): S

Escolha [P]ar ou [Í]mpar: Í
Escolha um número entre 1 e 10: 10

Você escolheu 10 e o computador escolheu 10.
Você perdeu! A soma é 20

Deseja jogar novamente? (s/n): n
Obrigado por jogar! Até a próxima.
```



© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1. Conhecimento em estrutura de loop em Python.
2. Aplicação do controle de fluxo.
3. Aplicação e revisão das estruturas de loop (for e while).

Saiba mais

Veja outras formas de programar o jogo Par ou Ímpar em Python:

CÓDIGO PYTHON. *Simples jogo do par ou ímpar*, 1 out. 2022. Disponível em: https://codigopython.blogs.sapo.pt/simples-jogo-do-par-ou-impar-210770. Acesso em: 28 fev. 2024.

Aprenda a criar um jogo de adivinhação em Python:

ALURA. *Python: começando com a linguagem*. Disponível em: https://cursos.alura.com.br/course/python-introducao-a-linguagem/task/24543. Acesso em: 28 fev. 2024.

Aprenda a fazer o Jogo da Forca em Python:

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

Referências da aula

Identidade visual: Imagens © Getty Images.

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

S10 – Aula 1 – Registro

Programar o jogo Par ou Ímpar

Seguindo as instruções da brincadeira Par ou Ímpar, crie em grupo um programa que simule a partida entre você e o computador.

Obs.: para simular a jogada do computador, você pode usar o código abaixo.

```
:  1 import random
   2 random.randint(1, 10)
   3

: 3
```

Dica

A biblioteca **random** é útil para gerar números aleatórios. Exemplo: o comando random.randint(1,10) escolhe um número aleatório entre 1 e 10.

Veja abaixo uma sugestão para a saída do programa:

```
Bem-vindo ao Jogo Par ou Ímpar!

Escolha [P]ar ou [Í]mpar: P
Escolha um número entre 1 e 10: 3

Você escolheu 3 e o computador escolheu 3.
Você ganhou! A soma é 6

Deseja jogar novamente? (s/n): S

Escolha [P]ar ou [Í]mpar: Í
Escolha um número entre 1 e 10: 10

Você escolheu 10 e o computador escolheu 10.
Você perdeu! A soma é 20

Deseja jogar novamente? (s/n): n
Obrigado por jogar! Até a próxima.
```

Instruções de entrega

Ao finalizar a atividade, envie para o Ambiente Virtual de Aprendizagem (AVA).

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Programação aplicada à Ciência de Dados

Estrutura de controle de fluxo

Aula 2

Código da aula: [DADOS]ANO1C2B2S10A2

Secretaria da Educação
SÃO PAULO
GOVERNO DO ESTADO

Exposição

Objetivo da aula

- Conhecer os conceitos de função.

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 baseadas em evidências.

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

Calculadora

Você se lembra de como fazer uma calculadora em Python?

```python
print("Essa é uma calculadora!")

while True:
    print("\nEscolha uma operação:")
    print("1. Somar")
    print("2. Subtrair")
    print("3. Multiplicar")
    print("4. Dividir")
    print("5. Sair")

    escolha = input("Digite o número da operação desejada: ")

    if escolha == '5':
        print("Obrigado por usar a calculadora. Até a próxima!")
        break

    if escolha in ['1', '2', '3', '4']:
        numero1 = float(input("Digite o primeiro número: "))
        numero2 = float(input("Digite o segundo número: "))

        if escolha == '1':
            resultado = numero1 + numero2
            print("Resultado: ", resultado)
        elif escolha == '2':
            resultado = numero1 - numero2
            print("Resultado: ", resultado)
        elif escolha == '3':
            resultado = numero1 * numero2
            print("Resultado: ", resultado)
        elif escolha == '4':
            if numero2 != 0:
                resultado = numero1 / numero2
                print("Resultado: ", resultado)
            else:
                print("Erro: Divisão por zero")
    else:
        print("Escolha inválida. Por favor, escolha uma operação válida.")
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Função

Uma função é um bloco de código reutilizável que realiza uma tarefa específica. Ela é definida usando a palavra-chave **def**, seguida pelo nome da função e seus parâmetros.

Assim como em uma receita de bolo, uma função em Python contém instruções para atingir um objetivo. Os ingredientes (parâmetros) são fornecidos à função, que os processa e entrega o resultado (bolo pronto), tornando o código mais organizado e compreensível.

Exposição

Razões para usar funções

1. Reutilização de código:
Evita repetições, escrevendo um comando uma vez e utilizando várias vezes.

2. Organização e modularidade:
Divide o código em partes menores e mais gerenciáveis.

3. Facilita a compreensão do código:
Torna o código mais legível e compreensível, especialmente para quem não escreveu o código original.

4. Revisão e depuração:
O uso de funções facilita a revisão e a realização de testes no código, melhorando sua qualidade final.

Tome nota

Utilizar funções não apenas economiza tempo, mas também torna a manutenção do código mais fácil e colaborativa.

Exposição

Sintaxe básica de funções

```
def nome_da_funcao(parametros):
    # Corpo da função
    # Realiza alguma tarefa
    return resultado
```

- **def:** palavra-chave para definir uma função.
- **nome_da_funcao:** identificador da função.
- **parâmetros:** variáveis que a função aceita como entrada.
- ***return*:** palavra-chave para retornar um valor da função (opcional).

A sintaxe básica de uma função inclui a definição, o nome da função, seus parâmetros e o bloco de código interno. O retorno é opcional, dependendo se a função produz um resultado.

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Exemplo

- Criamos a função **saudacao** que aceita um parâmetro **nome**.
- Dentro da função, geramos uma mensagem de saudação.
- Chamamos a função com o nome "Alice" e imprimimos o resultado.

Neste exemplo, a função **saudacao** recebe um nome como parâmetro e retorna uma mensagem de saudação personalizada. A chamada da função é feita passando o nome desejado, e o resultado é impresso.

```
1 def saudacao(nome):
2     mensagem = f"Olá, {nome}! Bem-vindo à nossa aula."
3     return mensagem
4 
5 # Chamando a função
6 resultado = saudacao("Alice")
7 print(resultado)
```

Olá, Alice! Bem-vindo(a) à nossa aula.

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Parâmetros e argumentos

```
1  def multiplicar(x, y):
2      resultado = x * y
3      return resultado
4
5  # Chamando a função com argumentos
6  resultado_final = multiplicar(5, 3)
7  print(resultado_final)
```

```
15
```

Parâmetros: são as variáveis listadas na definição da função.

Argumentos: são os valores passados para a função durante sua chamada.

Os parâmetros *x* e y da função **multiplicar** recebem os argumentos 5 e 3, respectivamente, durante a chamada da função.

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Retorno de funções

```python
def elevar_ao_quadrado(numero):
    resultado = numero ** 2
    return resultado

# Chamando a função e armazenando o resultado
quadrado_de_5 = elevar_ao_quadrado(5)
```

- Utilizamos a palavra-chave ***return*** para retornar um valor da função.
- O valor retornado pode ser armazenado em uma variável.

Nesse exemplo, a função **elevar_ao_quadrado** recebe um número como parâmetro, calcula o quadrado desse número e retorna o resultado. O valor retornado (25, no caso) pode ser usado em outras partes do programa.

Exposição

Funções com múltiplos parâmetros

```python
def calcular_media(nota1, nota2, nota3):
    media = (nota1 + nota2 + nota3) / 3
    return media

# Chamando a função com múltiplos parâmetros
media_aluno = calcular_media(8, 9, 7)
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

- Funções podem ter mais de um parâmetro.
- Ao utilizar vários parâmetros, é possível realizar várias operações em um único chamado de função, economizando a escrita de código redundante.
- Isso proporciona maior flexibilidade na manipulação de dados.

A função **calcular_media** recebe três notas como parâmetros e calcula a média. Essa flexibilidade permite adaptar a função para diferentes cenários.

Exposição

Escopo

O "escopo" em programação refere-se ao contexto em que as variáveis são definidas e podem ser acessadas. Em outras palavras, o escopo determina a visibilidade e a disponibilidade das variáveis em diferentes partes de um programa. Existem dois tipos principais de escopo: escopo local e escopo global.

Exposição

Escopo local

Variáveis definidas dentro de uma função ou bloco de código têm um escopo local. Elas só podem ser acessadas dentro desse contexto específico. Quando a função é encerrada, as variáveis locais deixam de existir.

```python
def exemplo():
    variavel_local = 10
    print(variavel_local)

exemplo()  # Saída: 10
print(variavel_local)  # Erro! A variável não está disponível fora da função.
```

```
10

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 6
      3     print(variavel_local)
      5 exemplo()  # Saída: 10
----> 6 print(variavel_local)  # Erro! A variável não está disponível fora da função.

NameError: name 'variavel_local' is not defined
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Escopo global

Variáveis definidas fora de qualquer função ou bloco de código têm um escopo global. Elas são acessíveis em todo o programa. No entanto, é importante evitar o uso excessivo de variáveis globais para manter a clareza do código e evitar dependências complexas.

```python
variavel_global = 20

def exemplo():
    print(variavel_global)

exemplo()  # Saída: 20
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Registro

Vamos fazer um ***quiz***

Qual das seguintes afirmações descreve corretamente uma função em Python?

- Um tipo de loop que repete um bloco de código várias vezes.
- É um conjunto de variáveis.
- É um bloco de código reutilizável que realiza uma tarefa específica.
- É uma expressão condicional que toma decisões no programa.

Registro

Vamos fazer um ***quiz***

Qual das seguintes afirmações descreve corretamente uma função em Python?

- ✗ Um tipo de loop que repete um bloco de código várias vezes.
- ✗ É um conjunto de variáveis.
- ✓ É um bloco de código reutilizável que realiza uma tarefa específica.
- ✗ É uma expressão condicional que toma decisões no programa.

FEEDBACK GERAL DA ATIVIDADE

Uma função em Python é exatamente isso: um bloco de código reutilizável que executa uma tarefa específica. Isso promove a modularidade e facilita a manutenção do código.

Registro

Vamos fazer um ***quiz***

O que são parâmetros em uma função Python?

- Variáveis definidas dentro de uma função.
- Valores de entrada que uma função aceita durante sua chamada.
- Resultados produzidos por uma função.
- Constantes que não podem ser alteradas dentro de uma função.

Registro

Vamos fazer um ***quiz***

O que são parâmetros em uma função Python?

- ✗ Variáveis definidas dentro de uma função.
- ✓ Valores de entrada que uma função aceita durante sua chamada.
- ✗ Resultados produzidos por uma função.
- ✗ Constantes que não podem ser alteradas dentro de uma função.

FEEDBACK GERAL DA ATIVIDADE

Parâmetros em uma função são os valores fornecidos durante a chamada da função. Esses valores são utilizados pela função para realizar operações específicas.

Registro

Vamos fazer um ***quiz***

Por que usar funções em Python?

- Porque reduz a complexidade do código e torna-o mais difícil de entender.
- Porque funções são úteis apenas para programadores avançados.
- Porque funções não oferecem a capacidade de reutilização de código.
- Porque ajuda na reutilização de código e na organização, bem como facilita a compreensão do programa.

Registro

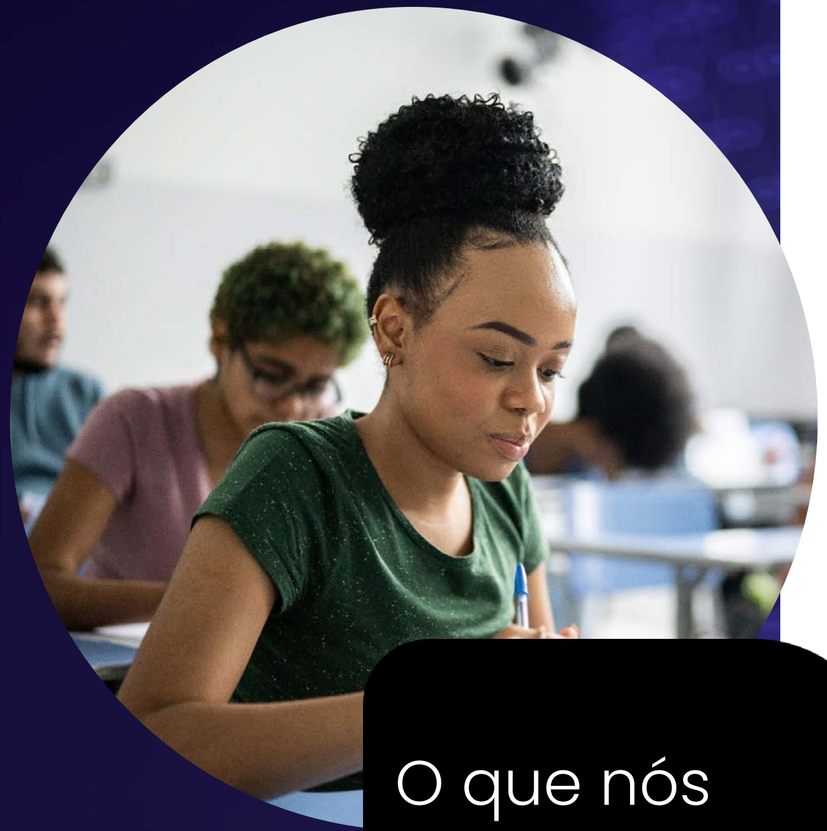
Vamos fazer um ***quiz***

Por que usar funções em Python?

- ✗ Porque reduz a complexidade do código e torna-o mais difícil de entender.
- ✗ Porque funções são úteis apenas para programadores avançados.
- ✗ Porque funções não oferecem a capacidade de reutilização de código.
- ✓ Porque ajuda na reutilização de código e na organização, bem como facilita a compreensão do programa.

FEEDBACK GERAL DA ATIVIDADE

O uso de funções em Python traz diversos benefícios, incluindo a reutilização de código (escrever uma vez, usar em vários lugares), organização (dividir o código em partes mais gerenciáveis) e facilitação da compreensão (código mais claro e legível).



© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1 Entendimento da definição de função.

2 Compreensão e aplicação do uso funções.

3 Conhecimento da sintaxe básica de funções.

Saiba mais

Entenda mais sobre *built-in functions* e funções:

ALURA. *Python para Data Science*: trabalhando com funções, estruturas de dados e exceções. Disponível em: https://cursos.alura.com.br/course/python-data-science-funcoes-estruturas-dados-excecoes/task/125896. Acesso em: 28 fev. 2024.

Referências da aula

Identidade visual: Imagens © Getty Images.

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

S10 – Aula 2 – Quiz

Condições de conclusão

Ver

Qual das seguintes afirmações descreve corretamente uma função em Python?

- ○ Um tipo de loop que repete um bloco de código várias vezes.
- ○ É um conjunto de variáveis.
- ○ É uma expressão condicional que toma decisões no programa.
- ○ É um bloco de código reutilizável que realiza uma tarefa específica.

O que são parâmetros em uma função Python?

- ○ Variáveis definidas dentro de uma função.
- ○ Constantes que não podem ser alteradas dentro de uma função.
- ○ Valores de entrada que uma função aceita durante sua chamada.
- ○ Resultados produzidos por uma função.

Por que usar funções em Python?

- ○ Porque ajuda na reutilização de código e na organização, bem como facilita a compreensão do programa.
- ○ Porque reduz a complexidade do código e torna-o mais difícil de entender.
- ○ Porque funções são úteis apenas para programadores avançados.
- ○ Porque funções não oferecem a capacidade de reutilização de código.

Educação Profissional Paulista

Técnico em Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Programação aplicada à Ciência de Dados

Estrutura de controle de fluxo

Aula 3

Código da aula: [DADOS]ANO1C2B2S10A3

Exposição

Objetivo da aula

- Aplicar os conceitos de funções 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 baseadas em evidências.

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

Função - Resumo

Uma função é um bloco de código reutilizável que realiza uma tarefa específica.

```
def nome_da_funcao(parametros):
    # Corpo da função
    # Realiza alguma tarefa
    return resultado
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook

- Promove a **reutilização** de código.
- Contribui para a **organização** e a **modularidade**.
- Facilita a **compreensão** do código.

Exposição

Criando funções

Outros exemplos de como criar uma função:

```
1  def minha_primeira_funcao():
2      print('Olá Mundo')
```

```
1  # chamar a função pelo nome para executá-la
2  minha_primeira_funcao()
```

```
Olá Mundo
```

```
1  def bom_dia():
2      print('Bom dia!')
```

```
1  bom_dia()
```

```
Bom dia!
```

```
1  def exemplo_funcao_simples():
2      print('criando uma função com o "def"')
```

```
1  exemplo_funcao_simples()
```

```
criando uma função com o "def"
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Parâmetros

Será que conseguimos personalizar?

```python
def funcao_ola(nome):
    print(f'Olá, meu nome é {nome}.')
```

```python
funcao_ola("Alice")
```

```
Olá, meu nome é Alice.
```

```python
funcao_ola("Renata")
```

```
Olá, meu nome é Renata.
```

```python
funcao_ola("Tiago")
```

```
Olá, meu nome é Tiago.
```

```python
def horario(hora):
    if hora < 12:
        print('Bom dia')
    elif hora < 18:
        print('Boa tarde')
    else:
        print('Boa noite')
```

```python
horario(10)
```

```
Bom dia
```

```python
horario(15)
```

```
Boa tarde
```

```python
horario(19)
```

```
Boa noite
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Ordem dos parâmetros

A ordem dos parâmetros importa?

```python
def exemplo(nome, idade):
    print(f"Nome: {nome}, Idade: {idade}")
```

```python
exemplo('Alice', 25)
```

```
Nome: Alice, Idade: 25
```

```python
exemplo(25, 'Alice')
```

```
Nome: 25, Idade: Alice
```

```python
exemplo(nome="Alice", idade=25)
```

```
Nome: Alice, Idade: 25
```

```python
exemplo(idade=25, nome="Alice")
```

```
Nome: Alice, Idade: 25
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Parâmetros e *return*

Qual é a diferença entre as funções?

```
1 def soma(a, b):
2     print(a + b)
```

```
1 soma(2, 9)
```

```
11
```

```
1 soma(7, 8)
```

```
15
```

```
1 soma(10, 15)
```

```
25
```

```
1 resultado = soma(10, 15)
2 print('O resultado é: ', resultado)
```

```
25
O resultado é:  None
```

```
1 def soma(a, b):
2     return a + b
```

```
1 soma(2, 9)
```

```
11
```

```
1 soma(7, 8)
```

```
15
```

```
1 soma(10, 15)
```

```
25
```

```
1 resultado = soma(10, 15)
2 print('O resultado é: ', resultado)
```

```
O resultado é:  25
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Parâmetros e *return*

```python
def soma(a, b):
    resultado = a + b
    print("print do resultado dentro da função: ", resultado) # apenas mostra na tela o valor de uma variável
    return resultado  # retorna um resultado/saída da função


resultado = soma(2, 3)

media = resultado / 2
print(f'A média é {media}')
```

```
print do resultado dentro da função:  5
A média é 2.5
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Chamar uma função como argumento

```
1  def soma_dois_valores(a, b):
2      resultado = a + b
3      return resultado
4
5  res = soma_dois_valores(2, 3)
6
7  def media_dois_valores(resultado):
8      media = resultado / 2
9      return media
10
11 print(media_dois_valores(res))
```

```
2.5
```

```
1  media_dois_valores(soma_dois_valores(2,3))
```

```
2.5
```

Elaborado especialmente para o curso com a ferramenta Jupyter Notebook.

Exposição

Exercícios

1. Faça uma função que recebe um número e imprima seu **dobro**.

2. Faça uma função que recebe o valor do raio de um **círculo** e retorna o valor do comprimento de sua circunferência **C = 2*pi*r**.

3. Crie uma função chamada **concatenar_palavras**, que receba duas *strings* como parâmetros e retorna a concatenação dessas duas *strings*, separadas por um espaço.

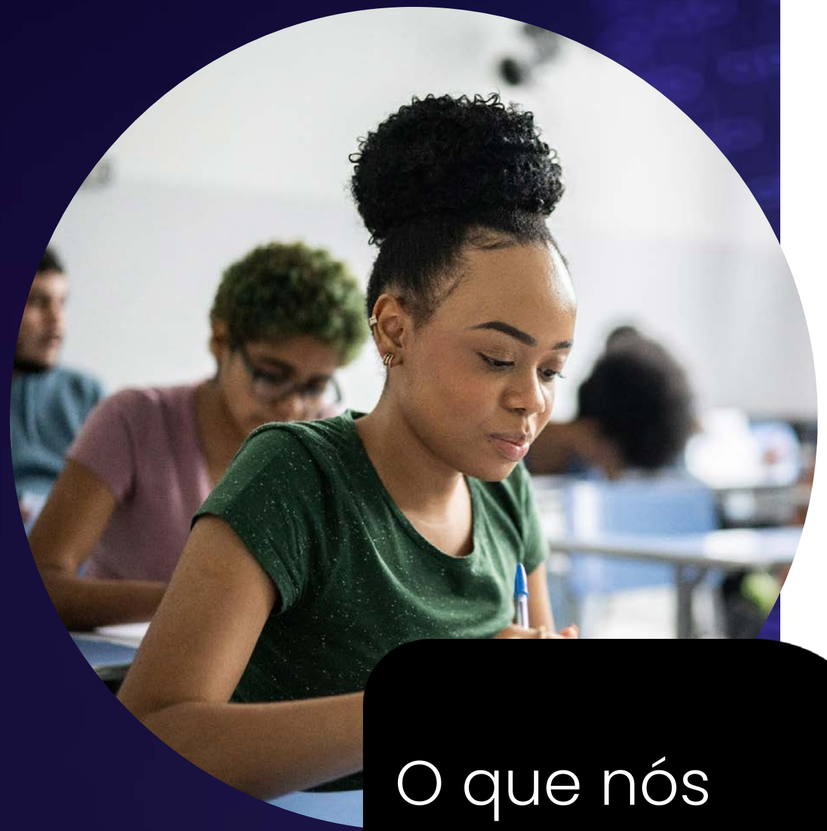
Exposição

Exercícios

4. Crie uma função chamada **verificar_par** que receba um número como parâmetro e retorne **True**, se o número for par, e **False**, em caso contrário.

5. Crie uma função chamada **calcular_media** que receba três números como parâmetros e retorne a **média aritmética** desses números.

6. Faça uma função para cada **operação matemática básica** (soma, subtração, multiplicação e divisão). As funções devem receber dois números e retornar o resultado da operação.



© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1 Aplicações práticas de funções em Python.

2 Resoluções de exercícios sobre funções.

Saiba mais

Entenda mais sobre *built-in functions* e funções:

ALURA. *Python para Data Science*: trabalhando com funções, estruturas de dados e exceções. Disponível em: https://cursos.alura.com.br/course/python-data-science-funcoes-estruturas-dados-excecoes/task/125896. Acesso em: 28 fev. 2024.

Referências da aula

Identidade visual: Imagens © Getty Images.

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

Programação aplicada à Ciência de Dados

Estrutura de controle de fluxo

Aula 4

Código da aula: [DADOS]ANO1C2B2S10A4

Exposição

Objetivo da aula

- Aplicar conceitos de funções.

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 baseadas em evidências.

Recursos didáticos

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

Duração da aula

50 minutos.

Exposição

Exercícios

1. Escreva uma função que retorne o **maior** de dois números.

2. Escreva uma função que receba dois números e retorne **True**, se o primeiro número for múltiplo do segundo.

3. Escreva uma função que receba o lado (l) de um quadrado e retorne sua área **(A = lado^2)**.

4. Escreva uma função que receba a base e a altura de um triângulo e retorne sua área **(A = (base x altura)/2)**.

Exposição

Exercícios

5. Faça uma função que recebe um **nome** e imprime “olá, [nome]”.

6. Faça uma função que recebe um **nome** e um **horário** e imprime “Bom dia, [nome]”, caso seja antes de 12h; “Boa Tarde, [nome]”, caso seja entre 12h e 18h; e “Boa noite, [nome]” caso seja após às 18h.

7. Faça uma função que recebe um **número** e retorna True, se ele for **par**, ou False, se ele for **ímpar**.

Exposição

Exercícios

8. Escreva uma função para validar uma variável ***string***. Essa função recebe como parâmetro a *string*, os números **mínimo** e **máximo** de caracteres. Retorne **verdadeiro**, se o tamanho da *string* estiver entre os valores de máximo e mínimo, e **falso** em caso contrário.

9. Escreva uma função que receba uma ***string*** e uma **lista**. A função deve comparar a *string* passada com os elementos da lista, também passada como **parâmetro**. Retorne verdadeiro, se a *string* for encontrada dentro da lista, e falso, em caso contrário.

Exemplo

Observe abaixo uma demonstração de lista:
exemplo de lista = ["Java", "C++", "Python", "JavaScript"]

Vamos fazer uma **atividade**

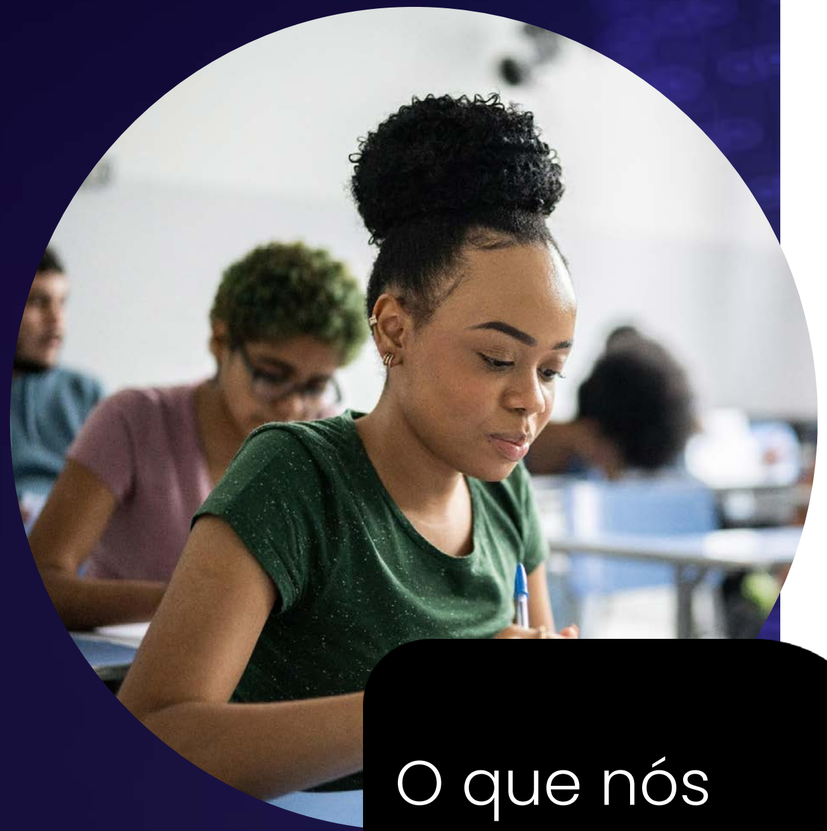
Calculadora

Transforme o código da imagem ao lado usando funções.

15 minutos

Em grupo

```python
print("Essa é uma calculadora!")

while True:
    print("\nEscolha uma operação:")
    print("1. Somar")
    print("2. Subtrair")
    print("3. Multiplicar")
    print("4. Dividir")
    print("5. Sair")

    escolha = input("Digite o número da operação desejada: ")

    if escolha == '5':
        print("Obrigado por usar a calculadora. Até a próxima!")
        break

    if escolha in ['1', '2', '3', '4']:
        numero1 = float(input("Digite o primeiro número: "))
        numero2 = float(input("Digite o segundo número: "))

        if escolha == '1':
            resultado = numero1 + numero2
            print("Resultado: ", resultado)
        elif escolha == '2':
            resultado = numero1 - numero2
            print("Resultado: ", resultado)
        elif escolha == '3':
            resultado = numero1 * numero2
            print("Resultado: ", resultado)
        elif escolha == '4':
            if numero2 != 0:
                resultado = numero1 / numero2
                print("Resultado: ", resultado)
            else:
                print("Erro: Divisão por zero")
    else:
        print("Escolha inválida. Por favor, escolha uma operação válida.")
```



© Getty Images

O que nós **aprendemos hoje?**

Hoje desenvolvemos:

1 Aplicações práticas de funções em Python.

2 Resoluções de exercícios sobre funções.

Saiba mais

Entenda mais sobre *built-in functions* e funções:

ALURA. *Python para Data Science*: trabalhando com funções, estruturas de dados e exceções. Disponível em: https://cursos.alura.com.br/course/python-data-science-funcoes-estruturas-dados-excecoes/task/125896. Acesso em: 28 fev. 2024.

Referências da aula

Identidade visual: Imagens © Getty Images.

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