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

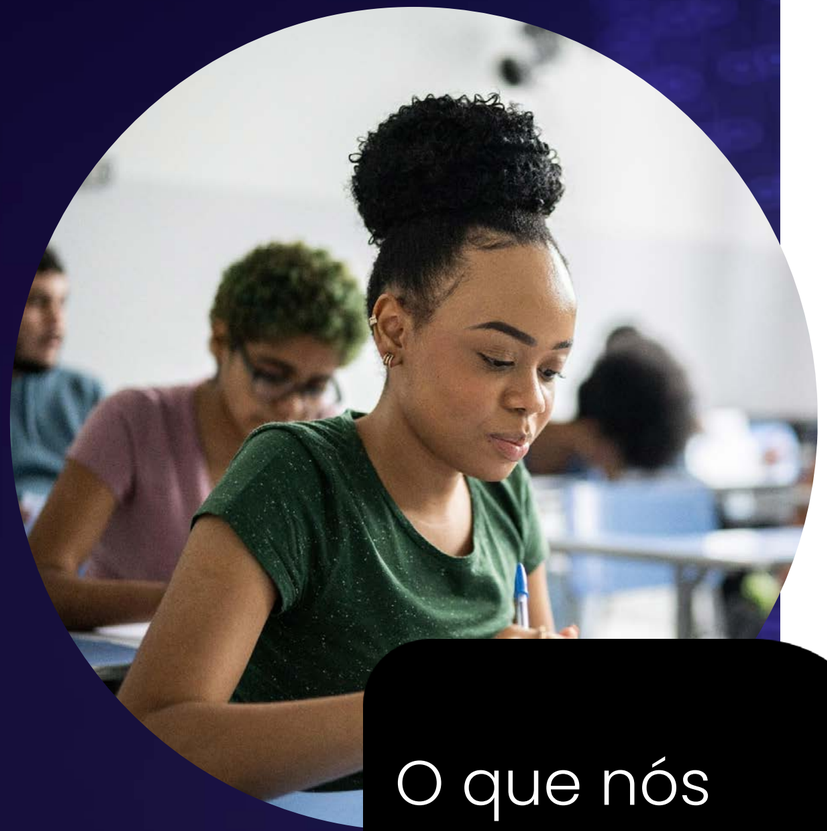
Vamos fazer um ***quiz***

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