Aula 1 – Funções

Objetivo da Aula

Conhecer e aplicar os conceitos de funções na linguagem de programação Python.

Apresentação

A construção de funções em Python é uma das principais formas de organizar e reutilizar código.

Uma das principais vantagens das funções em Python é que elas permitem a reutilização de código. Ao definir uma função para realizar uma tarefa específica, você pode chamá-la em diferentes partes do seu programa, sem precisar escrever o mesmo código várias vezes. Isso torna o seu código mais fácil de manter e atualizar.

As funções em Python também podem retornar valores, que podem ser usados em outras partes do seu programa. Para retornar um valor, você pode usar a palavra-chave "return" seguida do valor que deseja retornar. Se a função não tiver uma instrução "return", ela retornará "None" por padrão.

Além disso, as funções em Python podem ter parâmetros opcionais e valores-padrão para esses parâmetros. Isso permite que você crie funções mais flexíveis e genéricas, que possam ser usadas em diferentes contextos.

1. Funções em Python

É importante ter um conhecimento básico sobre funções em Python antes de aprender a criar classes, já que estas são uma forma mais avançada de estruturação de código e podem envolver o uso de funções.

Funções são uma das construções mais importantes da programação, pois permitem que você divida seu código em partes menores e mais gerenciáveis. Isso facilita a manutenção e o desenvolvimento de novos recursos em seu programa.

A instrução "def" é usada em Python para definir uma função. A palavra-chave "def" é seguida do nome da função, que deve seguir as mesmas regras de nomeação de variáveis em Python. Em seguida, entre parênteses, são definidos os parâmetros da função, que podem ser opcionais. A definição da função é concluída com dois pontos (:), indicando o início do bloco de código da função.

EXEMPLO

Vamos supor que você queira definir uma função simples em Python que some dois números. Você pode fazer isso usando a instrução "def" da seguinte maneira: nesse exemplo, a função somar recebe dois parâmetros, a e b, e retorna a soma desses dois números. O corpo da função começa na linha seguinte com a indentação e termina com a palavra-chave "return", que retorna o resultado da soma.

A instrução "def" é essencial em Python para definir funções que podem ser reutilizadas em diferentes partes do programa, permitindo que você divida seu código em partes menores e mais gerenciáveis.

Em Python, uma função é um bloco de código que executa uma tarefa específica e pode ser chamado de outros lugares no código. As funções ajudam a tornar o código mais modular e organizado, permitindo reutilizá-lo em diferentes partes do programa.

2. Funções Simples em Python

As funções simples em Python são definidas usando a instrução "def", seguida do nome da função, de uma lista de parâmetros (opcional) e do corpo da função. Aqui está um exemplo de uma função simples que imprime uma mensagem na tela:

```
1  def imprimir_mensagem():
2      print("Olá, mundo!")
```

Podemos chamar a função "imprimir_mensagem", em qualquer lugar do nosso código, usando o seguinte comando:

```
3
4  imprimir_mensagem()
```

Isso imprimirá a mensagem "Olá, mundo!" na tela.

3. Passagem de Parâmetros, Variáveis Globais e Locais

Em Python, podemos passar valores para uma função, usando parâmetros. Os parâmetros são listados entre os parênteses após o nome da função e separados por vírgulas. Aqui está um exemplo de uma função que recebe dois parâmetros:

```
def somar(a, b):
    resultado = a + b
    print(f"O resultado da soma é {resultado}")
```

Podemos chamar a função somar e passar dois valores como argumentos:

```
somar(2, 3)
```

Isso imprimirá a mensagem “O resultado da soma é 5”.

Também podemos passar outras estruturas de dados como parâmetros, como listas ou dicionários. Aqui está um exemplo de uma função que recebe uma lista como parâmetro e imprime seus elementos:

```
def imprimir_lista(lista):
  for elemento in lista:
      print(elemento)
```

Ainda, podemos chamar a função “imprimir_lista” e passar uma lista como argumento:

```
minha_lista = [“maçã”, “banana”, “laranja”]
imprimir_lista(minha_lista)
```

Isso imprimirá os elementos da lista na tela.

Em Python, as variáveis podem ser definidas dentro ou fora das funções. As variáveis definidas dentro de uma função são chamadas de “variáveis locais” e só existem dentro do escopo da função. As variáveis definidas fora de uma função são denominadas “variáveis globais” e podem ser acessadas de qualquer lugar do programa.

Aqui está um exemplo que mostra a diferença entre variáveis locais e globais:

```
variavel_global = “Esta é uma variável global”
def imprimir_variavel():
  variavel_local = “Esta é uma variável local”
  print(variavel_global)
  print(variavel_local)
imprimir_variavel()
print(variavel_global)
print(variavel_local) # Erro: variável local não definida
```

Isso imprimirá a variável global duas vezes (dentro e fora da função), mas produzirá um erro ao tentar imprimir a variável local fora da função.

4. Funções que Retornam Valor

Em Python, as funções que retornam um valor são aquelas que realizam uma operação e retornam o resultado para o local onde a função foi chamada. Para retornar um valor em Python, utilizamos a instrução “return”, seguida do valor a ser retornado.

Aqui está um exemplo de uma função que retorna o quadrado de um número:

```
def calcular_quadrado(numero):
    quadrado = numero ** 2
    return quadrado
```

Nesse caso, a função “calcular_quadrado” recebe um argumento “numero”, calcula o quadrado desse número e retorna o resultado. Podemos chamar essa função e armazenar seu resultado em uma variável:

```
resultado = calcular_quadrado(5)
print(resultado) # saída: 25
```

No exemplo acima, chamamos a função “calcular_quadrado” com o argumento 5 e armazenamos seu resultado na variável resultado. Em seguida, imprimimos o valor de resultado na tela, que será igual a 25.

Outro exemplo de função que retorna valor é aquela que retorna o maior valor em uma lista:

```
def encontrar_maior_valor(lista):
    maior = lista[0]
    for valor in lista:
        if valor > maior:
            maior = valor
    return maior
```

Nesse caso, a função “encontrar_maior_valor” recebe uma lista como argumento, percorre todos os valores da lista e retorna o maior valor encontrado. Podemos chamar essa função e armazenar seu resultado em uma variável:

```
numeros = [2, 8, 4, 10, 5]
maior_numero = encontrar_maior_valor(numeros)
print(maior_numero) # saída: 10
```

No exemplo acima, chamamos a função “encontrar_maior_valor”, com a lista “numeros” como argumento, e armazenamos seu resultado na variável “maior_numero”. Em seguida, imprimimos o valor de maior_numero na tela, que será igual a 10.

Além disso, as funções em Python podem retornar múltiplos valores usando tuplas. Por exemplo:

```
def calcular_quadrado_e_cubo(numero):
  quadrado = numero ** 2
  cubo = numero ** 3
  return quadrado, cubo
```

Nesse caso, a função “calcular_quadrado_e_cubo” recebe um argumento “numero”, calcula o quadrado e o cubo desse número e retorna os dois valores em uma tupla. Podemos chamar essa função e armazenar seus resultados em duas variáveis separadas:

```
resultado_quadrado, resultado_cubo = calcular_quadrado_e_cubo(3)
print(resultado_quadrado) # saída: 9
print(resultado_cubo) # saída: 27
```

No exemplo acima, chamamos a função “calcular_quadrado_e_cubo”, com o argumento 3, e armazenamos seus resultados nas variáveis “resultado_quadrado” e “resultado_cubo”. Em seguida, imprimimos os valores dessas variáveis na tela, os quais serão iguais a 9 e 27, respectivamente.

Considerações Finais da Aula

As funções em Python são blocos de código que realizam tarefas específicas e podem ser reutilizadas em diferentes partes de um programa.

Uma das principais vantagens de utilizar funções em Python é a modularidade. Ao dividir o código em funções, o programa se torna mais organizado e mais fácil de ser mantido. Além disso, as funções podem ser reutilizadas em diferentes partes do programa ou mesmo em outros programas.

Outra vantagem é a possibilidade de passar parâmetros para as funções. Isso permite que as funções recebam dados específicos para realizar suas tarefas, tornando-as mais flexíveis e genéricas. Os parâmetros podem ser de diferentes tipos, como strings, números ou, até mesmo, outras funções.

Outro cuidado é garantir que as funções tenham apenas uma responsabilidade. Funções com muitas tarefas diferentes podem se tornar difíceis de entender e modificar. Além disso, é recomendado documentar as funções com comentários, explicando seu funcionamento e parâmetros.

Materiais Complementares

Livro

Programação Em Python
2011, Vitor Amadeu Souza. Clube de Autores.
Esse livro apresenta informações sobre funções e como aplicá-las na linguagem Python.

Site

Pyquest/envExemplo/Lista04
2023, Alexandre N. Louzada.
Esse link permite o acesso a um repositório com várias listas de exemplos de programas em Python para auxiliar estudantes de programação.
Link para acesso: https://github.com/AlexandreLouzada/Pyquest/tree/master/envExemplo/Lista04 (acesso em 6 jun. 2023.)

Referências

ALVES, William P. ***Programação Python***: aprenda de forma rápida. Editora Saraiva, 2021.

PYTHON SOFTWARE FOUNDATION. Python Language Site. ***Documentation***, 2023. Página de documentação. Disponível em: https://docs.python.org/pt-br/3/tutorial/index.html. Acesso em: 8 mar. 2023.