Artigo

Funções em Python

Veja neste artigo como criar e utilizar funções na linguagem de programação Python.

Por que eu devo ler este artigo:

Funções são um recurso poderoso das linguagens de programação. Ao desenvolver uma aplicação utilizamos elas a todo mo

Ver mais

Marcado como lido

Anotar

Na programação, funções são blocos de código que realizam determinadas tarefas que normalmente precisam ser executadas diversas vezes dentro de uma

compreendendo sua definição e uso na prática.

Criando funções no Python

A sintaxe de uma função é definida por três partes: nome, parâmetros e corpo, o qual agrupa uma sequência de linhas que representa algum comportamento. No código abaixo, temos um exemplo de **declaração de função em Python**:

```
1 def hello(meu_nome):
2   print('Olá',meu_nome)
```

Essa função, de nome `hello`, tem como objetivo imprimir o nome que lhe é passado por parâmetro (também chamado de argumento). A palavra reservada `def`, na primeira linha, explicita a definição da função naquele ponto. Em seguida, entre parênteses, temos o parâmetro `meu_nome`. Ainda na mesma linha, observe a utilização dos dois pontos (:), que indicam que o código identado nas linhas abaixo faz parte da função que está sendo criada. Aqui, é importante ressaltar que, para respeitar a sintaxe da linguagem, a linha 2 está avançada em relação à linha 1.

Para executar a função, de forma semelhante ao que ocorre em outras linguagens, devemos simplesmente chamar seu nome e passar os parâmetros esperados entre parênteses, conforme o código a seguir.

vários argumentos, como no código abaixo:

```
def hello(meu_nome,idade):
    print('Olá',meu_nome,'\nSua idade é:',idade)
```

Agora, ao invocar essa função, também é necessário informar o segundo parâmetro, que representa a idade que será impressa após o nome:

```
>>> hello('Fabio',28)
Olá Fabio
Sua idade é: 28
```

Assim como podem receber valores de entrada, as funções também podem produzir valores de saída, provenientes de determinadas operações. Nos exemplos anteriores, apenas imprimimos um valor com a função `print`, sem retornar explicitamente um resultado. Já na **Listagem 1**, temos uma função que faz o cálculo do salário e retorna o valor a ser pago conforme o número de horas trabalhadas.

```
def calcular_pagamento(qtd_horas, valor_hora):
  horas = float(qtd_horas)
  taxa = float(valor_hora)
  if horas <= 40:
    salario=horas*taxa
  else:
    h_excd = horas - 40
    salario = 40*taxa+(h_excd*(1.5*taxa))
  return salario
```

Listagem 1. Calculando horas extras

esses valores são convertidos para o tipo float, pois eles serão recebidos como string por meio da instrução `input` .

Na quarta linha, verificamos se a quantidade de horas trabalhadas é menor ou igual a 40. Caso seja verdadeiro, na linha 5 calculamos o valor do salário apenas multiplicando a quantidade de horas pelo valor de cada hora trabalhada. Se a quantidade for maior que 40 (linha 6), adicionamos ao salário um valor adicional pelas horas extras. Por fim, na linha 9 retornamos o resultado do cálculo (contido na variável `salario`) com a instrução `return` .

No código abaixo, vemos como utilizar essa função, obtendo seu retorno e o imprimindo na tela posteriormente:

```
1 str_horas= input('Digite as horas: ')
2 str_taxa=input('Digite a taxa: ')
3 total_salario = calcular_pagamento(str_horas,str_taxa)
4 print('O valor de seus rendimentos é R$',total_salario)
```

Primeiramente, solicitamos do usuário as informações necessárias, que serão armazenadas como string e repassadas para a função (linhas 1 e 2). Em seguida, na linha 3, obtemos o resultado da função e o atribuímos à variável `total_salario` , que é impressa na linha 4.

A **Listagem 2** mostra a execução desse código em duas situações.

```
1 Digite as horas: 40
2 Digite a taxa: 20
3 O valor de seus rendimentos é:  800.0
4 >>>
```

Nesse caso, definimos explicitamente que a função deve retornar um determinado resultado por meio da instrução return. Caso isso não seja feito, o valor padrão retornado será None, equivalente ao null, void ou nil encontrado em outras linguagens.

Parâmetros nomeados

As funções em Python tem suporte a parâmetros nomeados. O exemplo a seguir mostra um caso onde podemos usar nomes nos parâmetros da função.

```
1 def calculo_imc(peso, altura):
2     print(peso / altura ** 2)
3
4 calculo_imc(75, 1.68)
```

Listagem 3. Função com parâmetros

Observe que quando chamamos a função calculo_imc, não há uma identificação do que cada valor representa dentro daquela função. Nesse mesmo exemplo usando essa funcionalidade, conseguimos ver melhor como podemos dar nome aos parâmetros.

```
1 def calculo_imc(peso, altura):
2     print(peso / altura ** 2)
3
4 calculo_imc(peso = 75, altura = 1.68)
```

Listagem 4. Função com parâmetros nomeados

Saiba mais: **Curso de Python**

Funções builtin no Python

A biblioteca do Python contém vários componentes embutidos, que podem ser utilizados em qualquer parte do código sem a necessidade de um import. Um exemplo disso é a função `max()`, que retorna o maior elemento de uma lista que lhe é passada por parâmetro. No código abaixo,

```
4 | print(maior_letra)
```

No primeiro exemplo, a variável `maior_numero` receberá o valor 3, por ele ser o maior elemento passado por parâmetro. Já no segundo, a variável `maior_letra` receberá a letra “c”, pois seu valor na tabela ASCII é superior ao das letras “a” e “b”.

Outros exemplos de funções builtin são: `input()`, que recebe o valor que é digitado pelo usuário; e `float()` e `int()`, utilizadas quando se faz necessário converter valores para float e int, respectivamente.

Para outras funções disponíveis em módulos, como é o caso daquelas presentes no módulo de matemática, é necessário que seja realizada a importação desse módulo, através do comando `import`, antes de utilizá-las. A seguir temos um exemplo disso:

```
1 | import math
2 | exponencial = math.exp(3)
3 | print(exponencial)
```

Nesse caso, a função `exp()`, do módulo math, só poderá ser utilizada após a importação dele, na linha 1.

Tecnologias:

Python

Por Pablo

Em 2019

Suporte ao aluno - Tire a sua dúvida.