Funções

Instrutor: Tarik Ponciano

Senac Fecomércio Sesc

Links da Disciplina

1. Discord: https://discord.gg/wt5CVZZWJs

2. Drive: https://drive.google.com/drive/folders/1hOl0DaPeAor7gnhKBUIlZ5n8lLRDNvUY?usp=sharing

3. Github: https://github.com/TarikPonciano/Programador-de-Sistema-SENAC

Funções – O que é

Na programação, funções são blocos de código que realizam determinadas tarefas que normalmente precisam ser executadas diversas vezes dentro de uma aplicação. Quando surge essa necessidade, para que várias instruções não precisem ser repetidas, elas são agrupadas em uma função, à qual é dado um nome e que poderá ser chamada/executada em diferentes partes do programa.

Funções – Declaração 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 indentado 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.

Funções - Execução

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.

```
1 >>> hello('Fabio')
2 Olá Fabio
3 >>> meu_nome
4 'Fabio'
```

Funções

Caso seja necessário, também é possível definir funções com nenhum ou vários argumentos, como no código abaixo:

```
1  def hello(meu_nome,idade):
2      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:

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

Funções

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. Abaixo temos uma função que faz o cálculo do salário e retorna o valor a ser pago conforme o número de horas trabalhadas.

```
1  def calcular_pagamento(qtd_horas, valor_hora):
2    horas = float(qtd_horas)
3    taxa = float(valor_hora)
4    if horas <= 40:
5      salario=horas*taxa
6    else:
7      h_excd = horas - 40
8      salario = 40*taxa+(h_excd*(1.5*taxa))
9    return salario
```

Funções

```
1  def calcular_pagamento(qtd_horas, valor_hora):
2    horas = float(qtd_horas)
3    taxa = float(valor_hora)
4    if horas <= 40:
5      salario=horas*taxa
6    else:
7      h_excd = horas - 40
8      salario = 40*taxa+(h_excd*(1.5*taxa))
9    return salario
```

Na linha 1, a função `calcular_pagamento()` recebe dois parâmetros, `qtd_horas` e `valor_hora`, que representam, respectivamente, a quantidade de horas a serem calculadas e o valor da hora. Nas linhas 2 e 3, esses valores são convertidos para o tipo float, pois eles serão recebidos como string por meio da instrução `input`.

Funções

```
1  def calcular_pagamento(qtd_horas, valor_hora):
2    horas = float(qtd_horas)
3    taxa = float(valor_hora)
4    if horas <= 40:
5      salario=horas*taxa
6    else:
7      h_excd = horas - 40
8      salario = 40*taxa+(h_excd*(1.5*taxa))
9    return salario
```

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`.

Funções

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.

Funções

```
1  Digite as horas: 40
2  Digite a taxa: 20
3  O valor de seus rendimentos é:  800.0
4  >>>
5  Digite as horas: 50
6  Digite a taxa: 20
7  1100.0
```

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.

Funções – 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)
```

Observe que quando chamamos a função calculo_imc, não há uma identificação do que cada valor representa dentro daquela função.

Funções – Parâmetros Nomeados

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)
```

Mesmo que venhamos a trocar a ordem dos argumentos na chamada da função, ela será executada corretamente da mesma forma, pois os parâmetros estão sendo identificados por um nome e não pela sua posição.

Exercícios

Python é uma linguagem de programação de alto nível, interpretada, orientada a objetos, funcional, de tipagem dinâmica e forte. Levando isso em conta, analise o código em Python abaixo.

```
def e(b):
    a = b*b
    return a
a = 10
e(a)
e(a)
print(e(a))
```

Nesse caso, ao executar o programa, o valor impresso será?

Senac Fecomércio Sesc

Exercícios

Escreva uma função que, dado o valor da conta de um restaurante, calcule e exiba a gorjeta do garçom, considerando 10% do valor da conta.

Crie uma função que permita contar o número de vezes que aparece uma letra em uma string.

Faça uma função que receba uma lista de números armazenados de forma crescente, e dois valores (limite inferior e limite superior), e exiba a sublista cujos elementos são maiores ou iguais ao limite inferior e menores ou iguais ao limite superior.

Exemplo:
lista inicial=[12,14,15,16,18,20,24,26,28,32,34,38]
limite inferior=13
limite superior = 26
lista exibida: [14,15,16,18,20,24,26]

https://wiki.python.org.br/ExerciciosFuncoes

Obrigado!!

Senac Fecomércio Sesc

Referências

1. https://www.devmedia.com.br/funcoes-em-python/37340
2. https://www.w3schools.com/python/python_functions.asp

Senac
Fecomércio
Sesc