centro universitário facvest
unifacvest

Fábrica de Software

Aula 09

Python

FUNÇÕES

Até aqui vimos como criar programas simples em Python. Em todos esses programas, salvamos os resultados das computações em variáveis ou então imprimimos os resultados desejados. Entretanto, existem situações nas quais uma dada porção do nosso programa será utilizada **muitas vezes**, em diferentes partes do programa, ou até mesmo por outros programas.

Em Python, uma **função** é uma sequência de comandos que executa alguma tarefa e que tem um nome. A sua principal finalidade é nos ajudar a organizar programas em pedaços que correspondam a como imaginamos uma solução do problema.

A sintaxe de uma **definição de função** é:

```
def NOME( PARÂMETROS ):
    COMANDOS
```

Os parâmetros especificam quais informações, se alguma, você deve providenciar para que a função possa ser usada. Outra forma de dizer isto é que os parâmetros especificam o que a função necessita para executar a sua tarefa.

```
# Exemplo (bem simples) de função em Python.
def mostraNumero(n):
    print("O valor do parâmetro é ", n)

mostraNumero(20)
```

```
O valor do parâmetro é  20
```

Observe que, nesse caso, a função chama-se mostraNumero. Essa função foi declarado com o comando def e possui um parãmetro n. No corpo da função temos um print. Esse print mostra a frase: “O valor do parãmetro é: ” e o ‘n’ que é o parâmetro passado. Para “chamar” essa função, colocamos o nome dela e o parâmetro entre parênteses. Veja como ela fica com outro número:

```
mostraNumero(500)
```

```
O valor do parâmetro é  500
```

Perceba que, como a função já foi criada, dessa vez só precisou “chamá-la”. Isso é conhecido como **chamada de função** ou **invocação da função**.

Pode existir qualquer número de comandos em uma função, mas eles tem que ter a mesma identação (4 espaços ou 1 tab) a partir do def.

OBS: Apesar de só estarmos falando sobre funções agora, nós já usamos funções várias vezes nas apostilas anteriores, como print(),len(),max(),etc.

Em vez de imprimir o parâmetro, podemos simplesmente retorná-lo, como abaixo:

```
# Exemplo de função que retorna um valor.
def f(n):
    return n
```

```
f(80)
```

```
80
```

Podemos usá-la atribuindo diretamente em uma variável:

```
resultado = f(10)
print("O valor do parâmetro é {}".format(resultado))
```

```
O valor do parâmetro é 10
```

Outro exemplo, usando a chamada diretamente do print:

```
# Outro exemplo de função que retorna um valor.
def quadrado(n):
    return n ** 2
```

```
print("O número {} elevado ao quadrado é {}".format(13, quadrado(13)))
```

```
O número 13 elevado ao quadrado é 169
```

Podemos usar também mais de um parâmetro na função:

```
# Exemplo de função que retorna mais de um valor.
def quociente_resto(x, y):
    quociente = x // y
    resto = x % y
    return (quociente, resto)
```

```
print("Quociente e resto: ", quociente_resto(18, 5))
```

```
Quociente e resto:  (3, 3)
```

```
print("Quociente e resto: ", quociente_resto(23, 3))
```

```
Quociente e resto:  (7, 2)
```

Algumas funções interessantes:

Função sleep ():

```
# Usando a função time
import time
print("Esse print aparece de forma imediata.")
time.sleep(5.2)
print("Paciência é um dom que eu não tenho. Esse aparece depois de 5.2 segundos.")
```

```
Esse print aparece de forma imediata.
Paciência é um dom que eu não tenho. Esse aparece depois de 5.2 segundos.
```

O método *sleep* suspende a execução pelo número de segundos informado em seu parâmetro. A sintaxe do método é muito simples, veja:

```
time.sleep(t)
```

Na sintaxe, ***t*** é a quantidade em segundos que você deseja que a execução seja interrompida. Antes de usá-la, você precisa importar a biblioteca time.

```
import time
```

Posteriormente, você pode utilizar o método *sleep* passando a quantidade de segundos que você deseja parar a execução:

```
time.sleep(5)
```

Função random()

O **Python random** é um módulo que faz parte da linguagem Python e é utilizado para gerar números pseudo-aleatórios. Também podemos selecionar os elementos de uma lista de forma aleatória ou exibir o seu resultado embaralhado. Portanto, é um recurso útil para ser utilizado em vários tipos de aplicações, como no desenvolvimento de jogos, em que precisamos construir alternativas diversificadas.

A linguagem python pode nos ajudar, pois possui um módulo - **random** que gera números aleatórios. Para isso, é necessário importar esse módulo:

```
import random()
```

A função **random()** gera números reais (float) entre 0 (incluído) e 1 (não incluído) que podem representar a probabilidade de um evento acontecer:

```
#importar a biblioteca e rodar a 1°vez
import random

x = random.random()
print(x)
```

```
0.6797150405104491
```

```
#Rodando pela 2° vez
x = random.random()
print(x)
```

```
0.8833827403257251
```

```
#Rodando pela 3° vez
x = random.random()
print(x)
```

```
0.5483064101093313
```

As funções **randrange()** e **randint()** geram aleatoriamente um número inteiro dentro de um intervalo dado pelo usuário. Semelhantemente a função **random()**, o limite inferior do intervalo é incluído, mas o superior não.

```
import random

x = random.randint(1,10)

print(x)
```

```
9
```

```
# retornará números entre 0 e 100 e que são divisíveis por 3.

random.randrange(0, 100, 3)
```

```
99
```

Algumas vezes, queremos que o gerador de número reproduza a sequência de números criada uma vez. Isso é obtido ao prover a mesma semente ambas as vezes ao gerador, usando a função `seed`.

```
import random

random.seed(121)

random.random()
```

```
0.08730148686581662
```

```
random.seed(121)

random.random()
```

```
0.08730148686581662
```

Função choice ()

choice() é uma função embutida na linguagem de programação Python que retorna um item aleatório de uma lista, tupla ou string. Para usá-la temos que importar random.

```
import random

list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(random.choice(list1))
```

```
3
```

```
print(random.choice(list1))
```

```
6
```

```
print(random.choice(list1))
```

```
5
```

Função shuffle():

A função aleatória shuffle () do Python embaralha String ou qualquer sequência. Misturar uma lista significa que estamos reorganizando a ordem dos elementos na lista aleatoriamente. Se você precisar executar o embaralhamento de acordo com seus requisitos, poderá passar o método aleatório personalizado no lugar de um argumento aleatório, que ditará a função shuffle () sobre como randomizar o item da lista.

```python
#Embaralhando lista
import random

lista = [20, 16, 10, 5];
random.shuffle(lista)
lista
```

```
[10, 5, 20, 16]
```

```python
lista2 = ['Ana','Maria','José','Diego']
random.shuffle(lista2)
lista2
```

```
['Ana', 'José', 'Maria', 'Diego']
```

EXERCÍCIOS

1) Peça ao usuário o nome de 5 pessoas e sorteie uma delas para um prêmio.

2) Peça ao usuário 5 nomes de pessoas e coloque-as em uma ordem embaralhada para uma apresentação de seminário.

3) Faça um jogo de adivinhação. O computador escolhe um número de 1 a 10 e o usuário tenta adivinhar. Ele então diz se o usuário acertou ou não através de uma mensagem. Coloque um delay para dar emoção.

4) Faça um jogo de pedra, papel e tesoura. Coloque a opção de jogar de novo ou desistir.

5) Vamos fazer uma lista de compras. O usuário deve indicar o produto eu preço dele. A cada produto o programa deve perguntar se ele deseja adicionar mais produtos ou finalizar. Quando ele indicar finalizar, o programa coloca o preço total, o produto mais barato e seu valor e o produto mais caro com seu valor.