Aula 6

Docupedia Export

Author:Goncalves Donathan (CtP/ETS)
Date:30-Aug-2023 21:30

Table of Contents

1 Exercício 1 - Jogo par ou ímpar

- Faça um jogo de par ou ímpar, onde o usuário informa o número que quer jogar e se deseja ser par ou ímpar;
- A máquina irá “escolher” um número aleatório e o programa irá retornar dizendo se o usuário ganhou ou perdeu a rodada;
- O jogo deve se repetir até o usuário perder;
- Mostre a quantidade de vitórias;

DICA 1: Utilize a biblioteca “random”;

OUTPUT:

```
---------------NOVO-JOGO--------------------

Digite um numero: 4

PAR OU IMPAR? impar
O computador jogou:  0
RESULTADO =  4

Você perdeu!
VITÓRIAS:  3

RECORD:  3

Digite 0 para sair 0
```

2 Correção

```python
import random

print ("Jogo par ou ímpar")
print ("-"*30)

while True:
    vitoria=0

    while True:
        jogador = int(input("Diga um valor: "))
        maquina = random.randint(0,10)
        resultado = jogador + maquina

        escolha=" "
        while escolha not in "PI":
            escolha = str(input("Diga Par ou Impar: ")).upper()[0]

        print("\nVocê escolheu {} e a máquina {}. Total de {}".format(jogador, maquina, resultado))

        if escolha=="P":
            if resultado%2==0:
                print("Você venceu!")
                vitoria += 1
            else:
                print("Você perdeu!")
                break

        elif escolha=="I":
            if resultado%2==1:
                print("Você venceu!")
                vitoria += 1
            else:
                print("Você perdeu!")
                break
```

3 Funções e passagem de Parâmetros

O conceito de **função** é um dos mais importantes na matemática. Em computação, uma função é uma sequência de instruções que computa um ou mais resultados que chamamos de parâmetros.
Sempre utilizaremos funções quando nosso programa repete por muitas vezes uma sequência de códigos, para economizar esforço e memória do programa criamos um bloco que contém a nossa sequência e sempre que quisermos executá-la chamamos a função pelo nome.
Para criar a função é usado o comando ***def*** e em seguida o nome da função.

```
def saudações():
    print("Bom dia!")
```

Com a nossa função criada, sempre que quisermos saudar alguém basta chamar a função **saudações**.

```
>>> saudações()
Bom dia!
```

A importância das funções fica mais clara e faz mais sentido quando o código a ser repetido é grande, portanto utilizando funções é possível diminuir o tamanho do programa.
Por exemplo, numa construção de um jogo você precisa usar várias vezes uma contagem regressiva de 3 segundos. Sem usar funções você teria várias linhas de código como a seguinte em seu programa:

```
for i in range(3,0,-1):
    print(i)
    time.sleep(1)
```

Porém você pode colocar estas linhas dentro de uma função **contagem**( ) e só chamá-la pelo nome.

```
def contagem():
    for i in range(3,0,-1):
        print(i)
        time.sleep(1)
```

```
contagem()
```

Podemos adicionar **parâmetros** à nossa função. Os parâmetros são os valores que damos para a função manipular, devem ser colocados dentro dos parênteses na declaração da função.
Vamos criar uma função que realiza a soma de dois números:

```
def soma(num1,num2):
    total = num1 + num2
    print(total)
```

Agora em qualquer lugar do programa pode ser chamada a função **soma** com valores diferentes para ser somados.

```
soma(1,2)
soma(13,4)
soma(5,7)
```

```
3
17
12
```

```
soma(num1 = 2, num2 = 10)
```

4 Exercício 2 - Função contador

- Crie um programa que contenha a **função contador( )** que receba 3 parâmetros: início, fim, e passo do contador;
- Realize a contagem de: **1 a 20 de 1 em 1**;
- **20 a 0 de 2 em 2**;
- **0 a 105 de 5 em 5**;
- **96 a 52 de 2 em 2**;
- **3 a 41 de 1 em 1;**
- **75 a 15 de 5 em 5;**
- **390 a 39 de 10 em 10;**
- **uma contagem personalizada;**
- DICA: Para gerar um delay, utilize a biblioteca “time”;

OUTPUT:

```
---------CONTADOR PASSO:  1  INICIO:  0 FIM:  10 ----------
0
1
2
3
4
5
6
7
8
9
10


---------CONTADOR PASSO:  2  INICIO:  20 FIM:  0 ----------
20
18
16
14
12
10
8
6
4
2
0



Digite o inicio do contador: 1

Digite o final do contador: 3

Digite o passo do contador: 1
---------CONTADOR PASSO:  1  INICIO:  1 FIM:  3 ----------
1
2
3
```

5 Correção

```python
import time
lista=[]
def contador(inicio,fim,passo):
    printinicio=inicio
    if inicio< fim:
        lista.clear()
        while inicio <= fim:
            lista.append(inicio)
            inicio+=passo
    elif inicio > fim:
        lista.clear()
        while inicio >= fim:
            lista.append(inicio)
            inicio-=passo
    print("---------CONTADOR PASSO: ",passo," INICIO: ",printinicio,"FIM: ",fim,"----------")
    printar()
    print("\n\n")
def printar():
    for i in range(len(lista)):
        print(lista[i])
        time.sleep(0.5)


contador(0,10,1)
contador(20,0,2)
a=int(input("Digite o inicio do contador: "))
b=int(input("Digite o final do contador: "))
c=int(input("Digite o passo do contador: "))
contador(a,b,c)
```

As variáveis criadas dentro de funções são chamadas **variáveis locais** e elas só existem na execução das funções. Se você tentar acessá-las fora da função o programa não vai encontrar.
As variáveis criadas fora das funções são chamadas **variáveis globais**, elas podem ser acessadas dentro e fora de qualquer função, para utilizar uma variável global dentro de uma função utiliza-se o comando **global**.

```
x = 5
def funcao():
    x = 2
    print(x)
funcao()
print(x)
```

```
2
5
```

```
x = 5
def funcao():
    global x
    x = 2
    print(x)
funcao()
print(x)
```

```
2
2
```

Cuidado! Nomear as variáveis do jeito acima pode ser prejudicial para compreensão e manutenção do programa.

Vamos utilizar a instrução **return** para fazer com que a função retorne algum valor. Isso quer dizer que podemos atribuir a uma variável o valor obtido da função. Por exemplo, vamos criar uma função que calcula a velocidade média a partir de tempo e distância:

```
def velocidade(distancia,tempo):
    v = distancia/tempo
    print(v)
```

Da maneira como está construída, a função mostra a velocidade na tela, porém não salva este valor em lugar nenhum!

```
def velocidade(distancia,tempo):
    v = distancia/tempo
    return v
```

```
resultado = velocidade(100,2)
print(resultado)
```

Utilizando **return** podemos atribuir o valor calculado a uma variável, agora conseguimos guardar este valor na memória.

Um parâmetro **default** é um parâmetro com um valor pré-definido, isto é, se o usuário não passar nenhum valor para este parâmetro, ele assume o valor pré-definido.
Os argumentos **default** sempre devem aparecer depois que todos os argumentos posicionais (aqueles que dependem só da posição).

```
def ola(nome='estranho'):
    print("Olá, {}".format(nome))
```

```
ola()
ola('Jorge')
```

```
Olá, estranho
Olá, Jorge
```

Se quisermos retornar mais de um valor podemos adicionar mais elementos ao **return**. Isso retorna uma tupla com os valores.

```
def calculadora(x,y):
    return x+y,x-y
```

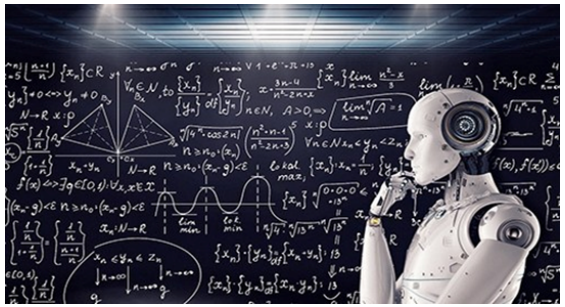
Podemos retornar um dicionário, e usar um laço **for** para imprimir os resultados.

```
def calculadora(x,y):
    return {'soma':x+y, 'subtração':x-y}
```

```
resultados = calculadora(3,4)
for i in resultados:
    print('{} : {}'.format(i,resultados[i]))
```

6 Exercício 3 - Função de operações

- Crie uma função **operacoes( )** que recebe um número do usuário, e retorna as seguintes operações em forma de dicionário:
  - Fatorial do número;
  - Inverso do número;
  - O número ao quadrado;
  - A raiz quadrada do número.
  - O usuário pode realizar essa tarefa quantas vezes quiser



OUTPUT:

```
Digite um valor: 10

Raiz quadrada: 3.1622776601683795

Quadrado: 100

Inverso: 0.1

Fatorial: 3628800
```

7 Correção:

```python
def op(x):
    fat=1
    for i in range(1,x+1):
        fat=fat*i
    return {'raiz quadrada': x**0.5, 'quadrado': x**2, 'inverso': x**-1, 'fatorial': fat}

a=int(input("Digite um valor: "))
resultados=op(a)
for j in resultados:
    print("{}: {}".format(j,resultados[j]))
```

8 Exercício 4 - Função lista ordenada

- Crie uma função com 3 parâmetros: limite inferior, limite superior, e tamanho da lista;
- Preencha a lista com números aleatórios dentro dos limites estabelecidos nos parâmetros;
- Ordene a lista em ordem crescente;
- NÃO UTILIZAR A FUNÇÃO “sort( )”
- DICA: Utilize a biblioteca “random”;

```
>>> import random
>>> help(random)
```

```
randint(self, a, b)
    Return random integer in range [a, b], including both end points.
```

OUTPUT:

```
Digite o limite inferior: 0

Digite o limite superior: 50

Digite o tamanho da lista: 10
LISTA:  [4, 13, 14, 16, 6, 11, 30, 0, 35, 4]
LISTA ORDENADA:  [0, 4, 4, 6, 11, 13, 14, 16, 30, 35]
```

9 Correção:

```
import random
def ordena(linf,lsup,tam):
    lista=[]
    for i in range(tam):
        lista.append(random.randint(linf,lsup))
    print(lista)
    for i in range(tam):
        for j in range(i+1,tam,1):
            if lista[j]<=lista[i]:
                lista[j],lista[i]=lista[i],lista[j]
    print(lista)
a=int(input("Digite o limite inferior: "))
b=int(input("Digite o limite superior: "))
c=int(input("Digite o tamanho da lista: "))
ordena(a,b,c)
```