INSTITUTO FEDERAL
Catarinense
Campus Blumenau

Algoritmos

Bacharelado em Ciência da Computação
Prof. Paulo César Rodacki Gomes

Lista de exercícios-Alg-05.1

Exercícios BÁSICOS sobre Funções - não precisa entregar

Exercícios: Funções (parte 1)

Para cada um dos exercícios, crie um arquivo fonte Python com o respectivo nome de acordo com as regras já determinadas nas listas de exercícios anteriores.

```
# OBSERVAÇÕES:
# Em cada exercício abaixo você deve criar e testar um função em Python
# Portanto, a estrutura de seu programa deve ter a DEFINIÇÃO da função e,
# posteriormente, a CHAMADA da função (a requisição para que ela seja executada
# Por exemplo:

# Definição de uma função:
def imprime_mensagem():
    print("Bom dia")

# Chamada da função
imprime_mensagem()

# PRESTE ATENÇÃO NA ESPECIFICAÇÃO DA FUNÇÃO no enunciado do exercício
# Isto inclui: O NOME DA FUNÇÃO, OS PARÂMETROS DE ENTRADA E O VALOR DE RETORNO
#
# EXEMPLO:
# Se o exercício pede: "faça uma função chamada "foo", que recebe dois
# inteiros e retorna uma string… etc… etc… etc…", a definição de sua função deve
# ser o mais parecida possivel com isso, por exemplo:

def foo(inteiro1, inteiro2):
    …
    …
    …
    return uma_string
```

Questões:

1. Escreva uma função Python chamada **imprime_nome(nome)**, que vai receber uma string e vai imprimir essa string 5 vezes.

2. Escreva uma função Python chamada **imprime_n_vezes(nome, n)**, que vai receber uma string e um numero inteiro n, e vai imprimir essa string n vezes.

3. Escreva uma função Python chamada **quociente(x, y)**, que vai receber receber 2 números **reais** x e y, e vai retornar o valor da divisão de x por y.

4. Escreva uma função Python chamada **quadrado(x)**, que vai receber receber um número real x, e vai retornar o valor de x ao quadrado.

5. Escreva uma função Python chamada **potencia(x, y),** que vai receber receber 2 números reais x e y, e vai retornar o valor de x elevado a y. Obs.: faça a operação de potenciação usando laço de repetição com comandos for ou while.

6. Faça uma função chamada **sorteia_dado()**, que deve retornar um número inteiro aleatório entre um 1 e 6 (inclusive). **Para gerar os números aleatórios, pesquise sobre a biblioteca random do Python**.

7. Usando a função da questão anterior, crie um novo programa que lance o dado 1 milhão de vezes. Conte e imprima quantas vezes cada número saiu. A probabilidade deu certo? Ou seja, a porcentagem dos números sorteados foi parecida?

8. Faça uma função que recebe um número inteiro n>0 e devolve o número de dígitos de n.

9. Faça uma função chamada **eh_bissexto(ano)** que recebe como parâmetro um inteiro positivo ano e devolve True se ano for bissexto, False em caso contrário.
   Obs.: Um ano é bissexto se (ano % 4 == 0 E (ano % 100 != 0 OU ano % 400 == 0)).)

10. Faça uma função chamada **conta_digitos(n, d)** que dados um inteiro n e um inteiro d, 0 < d <= 9, devolve quantas vezes o dígito d aparece em n.

11. Usando a função do item anterior, faça um programa que lê dois inteiros positivos a e b e responda se a é permutação de b.

    Um número a é dito permutação de um número b se os dígitos de a formam uma permutação dos dígitos de b.
    Exemplo: 5412434 é uma permutação de 4321445, mas não é uma permutação de 4312455.
    Obs.: Considere que o dígito 0 (zero) não aparece nos números.

12. Construa uma função chamada "**encaixa**" que, dados dois inteiros positivos a e b, verifica se b corresponde aos últimos dígitos de a.
    Exemplos:

a	b	
567890	890	=> encaixa
1243	1243	=> encaixa
2457	245	=> não encaixa
457	2457	=> não encaixa

13. Usando a função do exercício anterior, faça um programa que lê dois inteiros positivos a e b e verifica se o menor deles é segmento do outro.
    Exemplos:

A	b	
567890	678	=> b é segmento de a
1243	2212435	=> a é segmento de b
235	236	=> um não é segmento do outro