UTFPR - Universidade Tecnológica Federal do Paraná
DACOM - Departamento de Computação
BCC1001 :: Introdução à Programação

Lista de Exercícios no. 04 :: Funções

Instruções Gerais

- Os exercícios são de resolução individual.
- Sempre tente resolver os exercícios por conta própria, mesmo aqueles que já tenham sido feitos pelo professor em aula.
- Crie um arquivo (ex: lista.py) e faça cada exercício em uma ou mais funções. Faça chamadas de teste às funções dentro da função main(), conforme o exemplo no exercício 1.
- Utilize a extensão .py e o editor VS Code (ou outro de sua preferência).
- **Não é permitido o uso de recursos da linguagem ou bibliotecas que ainda não foram abordados na disciplina até o momento da publicação desta lista.**

1. Escreva uma função que calcula e devolve o somatório de um número: $\sum_{i=1}^{n} i$.

   Função: **`def summation(n: int) -> int`**

   OBS: Crie testes para todos os exercícios na função main():

```
def main():
    print(summation(10) == 55) # True
    print(summation(5)  == 15) # True
    print(summation(0)  == 0)  # True
    print(summation(-5) == 0)  # True

main()  # chamada da função main() para iniciar o programa
```

2. Utilizando a função **summation(n)** do exercício anterior, escreva uma função que devolve o produto (multiplicação) dos somatórios de dois números.

   Função: **`def summation_product(a: int, b: int) -> int`**

3. Escreva uma função que verifica se um dado número é primo. A função deve devolver **True,** caso o número seja primo, ou **False**, caso contrário.

   Função: **`is_prime(x: int) -> bool`**

4. Utilizando a função **is_prime(x)** do exercício anterior, escreva uma função que imprime os n primeiros números primos.

   Função: **`print_primes(n: int)`**

5. Escreva uma função que calcula e devolve a soma dos fatoriais até um dado número, $\sum_{i=1}^{n} i!$. Você pode escrever uma solução que utiliza 2 laços aninhados (laço dentro de laço).

   Função: **factorial_sum(n: int) -> int**

   Exemplo de uso:
   ```
   print(factorial_sum(5)) # deve calcular 1! + 2! + 3! + 4! + 5! = 153
   ```

6. Escreva uma nova versão da função do exercício anterior, agora utilizando apenas um único laço para realizar a operação solicitada na função. Dica: observe a representação dos cálculos.

   Função: **factorial_sum2(n: int) -> int**

   Exemplo de uso:
   ```
   print(factorial_sum2(5))
   ```

   ```
   1! + 2! + 3! + 4! + 5! = 153, é o mesmo que:

   1! = 1                = 1  +
   2! = 1 x 2            = 2  +
   3! = 1 x 2 x 3        = 6  +
   4! = 1 x 2 x 3 x 4    = 24 +
   5! = 1 x 2 x 3 x 4 x 5= 120
                           153
   ```

7. Escreva uma função que imprime N números inteiros aleatórios. A função receberá como parâmetro a quantidade de números (n) e os limites para sorteio (min e max). Os números devem ser randomizados entre [min, max]. Utilize a semente 1 antes de gerar os números, isto é, **random.seed(1)**.

   OBS: Será necessário utilizar a função **random.randint()** do módulo **random**. Para tanto, importe o módulo random com **import random**.

   Função: **print_random(n: int, min: int, max: int)**

   Exemplos de testes:
   ```
   print_random(5, -10, 10) # saída: -6 8 -8 -2 -7
   print_random(20, 0, 1)   # saída: 0 0 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 0 0 1
   ```

8. Escreva uma função que imprime uma linha com duas “pontas”. A função receberá como parâmetros: a largura da linha (n), o caractere de preenchimento da linha (fill) e o caractere das pontas da linha (edge).

   Função: **print_line(n: int, fill: str, edge: str)**

   Exemplo de uso:
   ```
   print_line(10, '-', '+') # Saída: +--------+
   ```

9. Utilizando a função do exercício anterior, escreva uma função que imprime uma caixa, recebendo como parâmetros:
    a. Largura e altura da caixa (width e height);
    b. O caractere de preenchimento da caixa (fill);
    c. O caractere das bordas da caixa (edge).

    Função: **print_box(height: int, width: int, fill: str, edge: str)**

    Exemplo de uso:
    ```
    print_box(5, 10, '#', 'o')
    ```

    ```
    o--------o
    |########|
    |########|
    |########|
    o--------o
    ```

10. Escreva uma função que imprime uma sequência de **n** números randômicos entre [left, right] (inclusivos). Ao final, a função deve devolver o maior e o menor números sorteados.

    Função: **print_random2(n: int, left: int, right: int) -> tuple[int, int]**

    Exemplo de uso:
    ```
    # imprime 50 números entre -500 e 500. Devolve o maior e o menor.
    min,max = print_random2(50, -500, 500)
    print(f'min: {min}, max: {max}')
    ```

11. Escreva uma função que imprime uma sequência de **n** números randômicos entre [left, right] (inclusivos). Ao final, a função deve devolver a contagem de pares e primos. Considere 0 (zero) como par. OBS: utilize a função **is_prime(x)** do exercício 3 para simplificar a verificação de número primo.

    Função: **rand_report(n: int, left: int, right: int) -> tuple[int, int]**

    Exemplo de uso:
    ```
    # imprime 50 números entre -500 e 500. Devolve contagem de pares e primos
    even,prime = rand_report(50, -500, 500)
    print(f'total: {n}, even: {even}, odd: {n-even}, primes: {prime}')
    ```