Universidade Federal do Ceará
Fundamentos de Programação
Prof. Rafael Ivo

Lista de exercícios 4 - Funções

1) Crie uma função **`dobro(n)`** que recebe como parâmetro um número inteiro e devolve seu dobro.

2) Crie uma função **`relogio_segundos(h,m,s)`** que receba três números inteiros como parâmetros, representando horas (**`h`**), minutos (**`m`**) e segundos (**`s`**), e os converta em segundos. Ex:
`relogio_segundos(2,40,10) = 9610` (2h, 40min e 10s correspondem a 9.610 segundos).

3) Escreva uma função **`somaAte(n)`** que retorna a soma de todos os números naturais até e incluindo o número ***`n`***. Não utilize nenhum método pronto da linguagem Python para realizar a soma automática. Ex:
`somaAte(10) = 1 + 2 + 3 + 4 + … + 10 = 55`

4) Crie uma função **`somaDivisores(n)`** que receba como parâmetro um valor inteiro positivo e retorne a soma dos divisores desse valor. Ex:
`somaDivisores(18) = 1 + 2 + 3 + 6 + 9 + 18 = 39`

5) Crie um programa que receba três valores **`l1`**, **`l2`** e **`l3`** (obrigatoriamente maiores do que zero), representando as medidas dos três lados de um triângulo. Seu programa deve conter duas funções:

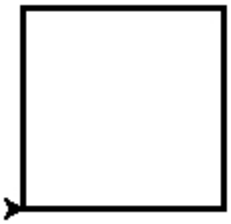
- a função **`trianguloValido(l1,l2,l3)`** deve retornar um valor booleano indicando se esses lados podem formar um triângulo válido (Dica matemática: um triângulo só pode ser formado por três lados se cada um dos lados for menor que a soma dos outros dois)
- a função **`trianguloTipo(l1,l2,l3)`** deve retornar uma string que indique o tipo do triângulo quanto aos seus lados:
  - *equilátero* para caso os três lados sejam iguais;
  - *isósceles* para caso dois lados sejam iguais e um diferente;
  - *escaleno* caso os três lados sejam diferentes.

Obs₁: A segunda função deve ser chamada apenas se a primeira determinar que os lados entrados forma um triângulo válido.
Obs₂: Nenhum print deve ser usado dentro das funções. Todas as mensagens exibidas na tela devem estar no corpo principal do programa.

6) Utilizando o módulo **turtle** de Python, crie as seguintes funções:
a) uma função **quadrado(c,l)** que desenha um quadrado de lado **l** na tela utilizando a caneta **c**.
b) uma função **quadradoLinha(c,l,n)** que desenha **n** quadrados de lado **l** espaçados por uma distância **l**
c) uma função **quadradoGrade(c,l,n)** que desenha uma grade de **n** linhas e **n** colunas de quadrados de lado **l** espaçados por uma distância **l**.
d) uma função **quadradoEspiral(c,l,n)** que desenha **n** quadrados de lado **l** rotacionados em torno da posição inicial

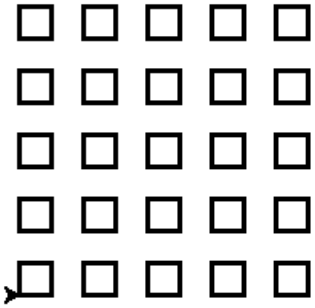
Exemplos:

quadrado(c,100)

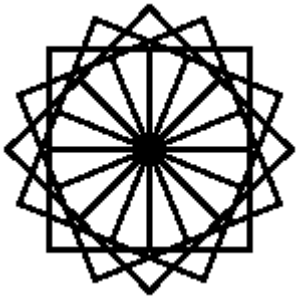


quadradoLinha(c,20,5)

quadradoGrade(c,20,5)



quadradoEspiral(c,50,16)

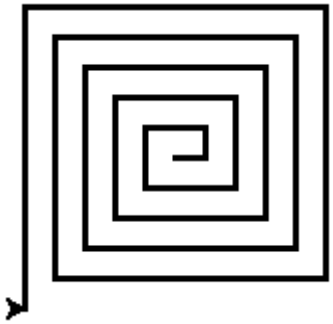


Atenção! Para todas as funções acima, siga as seguintes restrições:

- Antes de cada função, suponha que a caneta não esteja tocando o desenho.
- Nenhuma função deve utilizar posicionamento ou orientação fixa (ex: usar goto(x,y))
- A posição, orientação e estado da caneta ao final do desenho devem ser os mesmos após o término da função.

7) Escreva uma função **espiralQuadrada(c,l,n)** que desenhe uma espiral quadrada onde a distância entre as linhas seja igual a **l** e dê **n** voltas. Exemplo:

espiralQuadrada(c,15,5)



8) Escreva as seguintes funções:
a) a função **`circulo(c,r,cor)`** desenha um círculo de raio **`r`** utilizando a atual posição da caneta **`c`** como centro. A borda do círculo é preta e seu interior é definido pela string **`cor`**.
b) a função **`aneis(c,r1,r2,n,cor)`** desenha um conjunto de **`n`** círculos de raio **`r1`**. Cada círculo possui seu centro sobre um círculo de raio **`r2`** onde o centro é a posicão inicial da caneta.
c) a função **`alvo(c,r,n,cor1,cor2)`** desenha um conjunto de **`n`** círculos de cores alternadas entre **`cor1`** e **`cor2`**. Todos os círculos possuem o mesmo centro (a posição inicial da caneta). O raio do círculo mais interno é igual a **`r`** e os demais são iguais a múltiplos de **`r`**.

`circulo(c,100,"red")`	`aneis(c,20,100,32,"red")`	`alvo(c,10,10,"red","white")`