Computação I - Python
Laboratório 2

Para fixar os conceitos adquiridos na Aula 2, você deve agora definir e testar funções para alguns problemas aqui propostos. Já sabemos da existência de diferentes tipos de dados numéricos, que é possível usar funções previamente definidas para realizar parte do cálculo de uma nova função (uso articulado de funções), da existência de módulos do Python, que agrupam definições de funções que podem ser utilizadas no desenvolvimento de novas funções, e como usar argumentos default. Nos problemas propostos você deverá fazer uso destes conceitos adequadamente. É a sua oportunidade de exercitar os novos conhecimentos e fixá-los. Repare que agora você pode construir soluções para problemas mais complexos, mesmo não conhecendo muitos comandos do Python, pois pode usar funções já prontas que fazem coisas complexas (sem ter que se preocupar em como elas estão fazendo isso).

Seguindo com nossas boas práticas, para cada um dos exercícios a seguir:

- antes de começar a escrever código, faça o estudo do problema e o planejamento de sua solução.
- lembre de botar a **documentação** direitinho, dizendo o que a função faz, quais suas entradas e qual o **tipo de dado** de cada entrada, bem como do valor de retorno da função; por exemplo, se sua função recebe dois números inteiros, nos parâmetros chamados a e b e retorna a divisão deles (possivelmente um número fracionário):

```
'''Calcula e retorna a divisão de a por b;
   int, int -> float'''
```

- escolha **nomes elucidativos** para suas funções e parâmetros;
- pense em **valores de teste** relevantes para testar sua função. Ela tem alguma resposta esperada para valores negativos? Valores fracionários? Que tal testar também com valores no extremo do conjunto de dados de interesse da função (maiores valores esperados, menores valores esperados)?
- quando não estiver com dificuldade para entender algum erro de funcionamento ou resultado inadequado de sua função, não fique paralizado olhando para a tela! Pegue lápis e papel e recorra ao **teste de mesa**.
- **para fazer a entrega desta atividade prática, escreva suas funções no editor do IDLE, salvando todas em um único arquivo.**

Vamos lá!

---

1 Cálculos Algébricos

Esta seção é composta por problemas de aplicação direta de algum cálculo algébrico: como é o problema da média. São problemas simples, onde a identificação das entradas e saída da função, assim como o cálculo que deve ser feito é praticamente imediato.

1. Existem duas funções já definidas no Python, chamadas *max* e *min*, que retornam, respectivamente, o valor máximo e mínimo dentre os valores passados como parâmetros. Estas funções, peculiarmente, podem ter um número variável de entradas (isso é um recurso avançado que não vamos ver agora como é feito, mas já podemos usar funções que tem essa característica). Para a sequência seguinte de exercícios, você pode usar (onde achar necessário) as funções *max* e *min* do Python.

   (a) teste as funções *max* e *min* no console do Python, digitando, por exemplo:

   ```
   >>> max(3,2.8,3.9)

   >>> min(7,2,4,1,0)
   ```

   (b) Faça uma função que calcule e retorne a média de três números inteiros.

   (c) Faça uma função que retorne, dados três números, a diferença do **maior** deles com a média (obrigatoriamente use a função desenvolvida no item b).

   (d) Faça uma função que retorne, dados três números, a soma do menor deles com a média (obrigatoriamente use a função desenvolvida no item b).

Solution:

(a)

```
def mediaTresNumeros(a, b, c):
    ''' Dados três números 'a', 'b' e 'c', calcula a
    média aritmética entre eles
    float, float, float -> float
    Explicação: Basta somar os três números e dividir
    por 3, pela definição de média aritmética '''
    return (a + b + c)/3

def diferencaDoMaiorPelaMedia(a, b, c):
    ''' Dados três números 'a', 'b' e 'c', calcula a diferença
    entre o maior elemento e a média aritmética entre os três
    float, float, float, float -> float
    Explicação: Usa a função max para pegar o maior termo e
    subtrai da media obtida com a função do item a '''
    return max(a, b, c) - mediaTresNumeros(a, b, c)

def somaDoMenorComAMedia(a, b, c):
    ''' Dados três números 'a', 'b' e 'c', calcula a soma do
    menor elemento com a média aritmética entre os três
    float, float, float, float -> float
    Explicação: Usa a função min para pegar o menor termo e
    soma com a media obtida com a função do item a '''
    return min(a, b, c) + mediaTresNumeros(a, b, c)
```

2. O discriminante (também chamado de delta) de uma equação do segundo grau é a parte da fórmula de Bháskara na qual se deve calcular a raiz quadrada. Essa parte é representada pela letra grega Δ e pode ser encontrada por meio da seguinte equação: $\Delta = b^2 - 4 * a * c$. A fórmula de Bháskara, usada para o cálculo das raízes reais de uma equação de segundo grau é a seguinte:

$$x' = \frac{-b + \sqrt{\Delta}}{2 * a}$$

$$x'' = \frac{-b - \sqrt{\Delta}}{2 * a}$$

(a) faça uma função que dados os coeficientes a, b e c, calcula o discriminante de um polinômio do segundo grau.

(b) faça uma função que calcule a primeira raiz real de uma equação do segundo grau, dados seus coeficientes a, b e c.

(c) faça uma função que calcule a segunda raiz real de uma equação do segundo grau, dados seus coeficientes a, b e c.

Obs: Assuma que o usuário só vai passar valores que tornem o Δ positivo, ou seja, desconsiderar raízes complexas. Quando for escolher valores para testar sua função, esteja atento a isso! O que acontece se você testar com valores que tornam o delta negativo? **Obs2:** Use seu senso crítico ao testar a função, conferindo se os valores que a função está retornando realmente condizem com o resultado que você espera. Muita gente **erra essa questão porque não confere os resultados de seus próprios testes**, e perde a oportunidade de corrigir o erro antes de entregar a atividade (a vida do aprendiz de programador não é fácil).

Solution:

```
import math
    def primeiraRaizReal(a, b, c):
    ''' Dados os coeficientes de uma equação de segundo grau
    'a', 'b' e 'c' como em ax + bx + c, calcula a primeira raiz real
    float, float, float -> float
    Explicação: A primeira raiz real se calcula usando o menos
    na fórmula de Bháskara '''
    return (-b - math.sqrt(discriminante(a, b, c)))/(2*a)


def segundaRaizReal(a, b, c):
    ''' Dados os coeficientes de uma equação de segundo grau
    'a', 'b' e 'c' como em a*a + bx + c, calcula a segunda
    raiz real
    float, float, float -> float
    Explicação: A segunda raiz real se calcula usando o mais
    na fórmula de Bháskara '''
    return (-b + math.sqrt(discriminante(a, b, c)))/(2*a)
```

3. Faça uma função que calcule a soma de uma progressão aritmética dados o valor inicial (A_1), o valor final (A_n) e a razão (r).

(a) Primeiro, faça um estudo do problema para descobrir como é feito esse co cálculo ;-). Depois, planeje uma solução de forma que você escreva duas funções, descritas a seguir:

(b) uma para calcular o número de termos dados os valores inicial e final e a razão;

(c) outra para calcular a soma da PA dados os valores inicial, final e a razão.

Solution:

```
def quantidadeDeTermosPA(a1, an, r):
```

```
    ''' Dados o primeiro termo 'a1', o último termo 'an' e a
    razão 'r' de uma PA, determina a quantidade de termos
    float, float, float -> int
    Explicação: Pela fórmula geral, an = a1 + (n - 1)*r ==> n
    = (an - a1 + r)/r ==> (an - a1)/r + 1 '''
    return (an - a1)/r + 1

def somaDosTermosPA(a1, an, r):
    ''' Dados o primeiro termo 'a1', o último termo 'an' e a
    razão 'r' de uma PA, determina a soma de todos os termos
    float, float, float -> float
    Explicação: Pela fórmula da soma, Sn = (a1 + an) * n / 2 '''
    return ((a1 + an) * quantidadeDeTermosPA(a1, an, r))/2
```

2 Cálculos Geométricos

Esta seção aborda problemas puramente matemáticos, mas aplicados à geometria: aqui também a identificação das entradas e saídas da função são praticamente imediatos. Para realização dos cálculos basta lembrar das aulas de geometria.

4. Usando funções do módulo *math*, defina:
   (a) uma função que calcule a distância entre dois pontos em um plano dadas suas coordenadas.
   (b) uma função que calcule o perímetro de um triângulo reto dados os catetos. Use a função definida no item *a*.
   (c) uma função que calcule a soma do quadrado do seno com o quadrado do cosseno de um ângulo.

   Experimente com cada função no console fazendo pelo menos três exemplos com cada uma.

Solution:

(a)
```
import math
def distanciaPontos(x1, y1, x2, y2):
    ''' Dados dois pontos de coordenadas 'x1', 'y1' e 'x2',
    'y2', calcula a distância entre eles
    float, float -> float
    Explicação: A distância entre dois pontos nada mais é do
    que a hipotenusa de um triângulo de catetos x2 - x1 e y2 - y1 '''
    return hipotenusa(x2 - x1, y2 - y1)
```

(b)
```
import math
def perimetroTrianguloReto(b, c):
    ''' Dados os catetos 'b' e 'c' de um triângulo retângulo,
    calcula o perímetro do triângulo
    float, float -> float
    Explicação: O perimetro de um triângulo é a soma de seus
    três lados '''
    return hipotenusa(b, c) + b + c
```

(c)
```
import math

def identidadeTrigonometrica(angulo):
    ''' Dado um ângulo 'angulo', calcula a identidade
    trigonométrica com esse ângulo; float -> float
    Explicação: A identidade trigonométrica diz que
    sen(teta)^2 + cos(teta)^2 = 1, para qualquer teta '''
    return math.sin(angulo)**2 + math.cos(angulo)**2
```

5. Escreva uma função que calcule a área de um setor circular, dados o raio e o ângulo. Use um argumento *default* para o ângulo, de modo que se nenhum ângulo for informado, a função retorne a área do círculo inteiro.

Solution:

```
def areaSetorCircular(r, angulo = 360):
    ''' Dado o raio 'r' de uma circunferência e um ângulo
    'angulo', calcula a área do setor circular correspondente
    float, float -> float
    Explicação: A área de um setor circular é a área de uma
    fração da circunferência, determinada por angulo / 360 '''
    return (math.pi*r**2)/(360/angulo)
```