Computação I - Python
Laboratório 3

Para fixar os conceitos adquiridos na Aula 3, vamos definir e testar funções para mais alguns problemas propostos. Agora, além de sabermos trabalhar com tipos numéricos, sabemos que os valores envolvidos em um processamento podem também ser cadeias de caracteres (str) e booleanos (bool). E sabemos como executar algumas operações envolvendo valores desses tipos. Sabemos também o que são expressões booleanas e como usá-las para definir decisões que precisam ser tomadas durante a resolução de um problema. Nos problemas propostos você deverá fazer uso destes conceitos sempre que possível.

Continuamos a resolver problemas puramente matemáticos, como nas aulas anteriores, mas, como agora já possuímos mais recursos de programação (e tipos de dados com os quais podemos trabalhar), temos mais problemas de aplicações variadas. Algumas das soluções que você vai desenvolver aqui ainda serão construídas através de algum cálculo direto a partir das entradas da função. Outros requerem algoritmos ligeiramente mais complexos em suas soluções, incluindo a tomada de decisão em função dos valores das entradas (com o uso de estruturas condicionais). Conseguiremos resolver diversos problemas que não tínhamos como resolver antes.

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 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. Defina uma função que retorne o valor absoluto de um número fornecido. Observamos que Python provê uma função `abs` para esse fim. Faça a sua sem usá-la.
   Para refletir: Compare o comportamento da sua função e da função `abs` do Python para diferentes tipos de dados numéricos, incluindo complexos. Há diferenças? Lembre de fazer com que a descrição da sua função mostre para que tipos numéricos ela funciona.

Solution:

```
# Função que retorna o valor absoluto de um número
# float -> float
# 2 casos de teste necessários: um valor positivo e um valor negativo.
from math import abs
def absoluto1(numero):
    if numero < 0:
        return -numero
    return numero
```

2. Defina uma função que retorne quantas (uma, duas ou nenhuma) são as raízes reais de uma equação de segundo grau, dados os valores dos três coeficientes.
   Observação: Na aula 2 você já definiu uma função para calcular o discriminante da equação. Para aproveitá-la aqui, basta copiar o código correspondente neste novo arquivo. Ou, melhor ainda, você pode importar (usando o import) o seu próprio arquivo da aula anterior ;-)

Solution:

```
# Função que calcula o delta de uma equação do segundo grau
# float,float,float -> float
def delta(a,b,c):
    return b**2 - 4*a*c

# Função que retorna quantas são as raizes reais de uma equação do
# segundo grau dados os coeficientes.
# float,float,float -> int
# 3 casos de teste: valores de a,b,c que resultem em
# 1) delta positivo, 2) delta = 0, 3) delta negativo

def raizes(a,b,c):
    if delta(a,b,c) >0:
        return 2
    elif delta(a,b,c) == 0:
        return 1
    return 0
```

3. Imagine que você quer enviar uma mensagem de felicitações e gostaria de repetir essas felicitações muitas vezes na mensagem. Algo do tipo "Feliz aniversário!! Feliz aniversário!! Feliz aniversário!! ..." até encher a tela do seu amigo. Você pode escrever uma vez e ir copiando e colando. Aí você percebe, lá no meio do caminho, que digitou errado...Ai! Agora que você sabe usar Python para fazer o trabalho para você, faça uma função que receba como entrada um texto e o número `n` de repetições desejado e retorne uma sequência de caracteres composta por `n` repetições deste texto.

Solution:

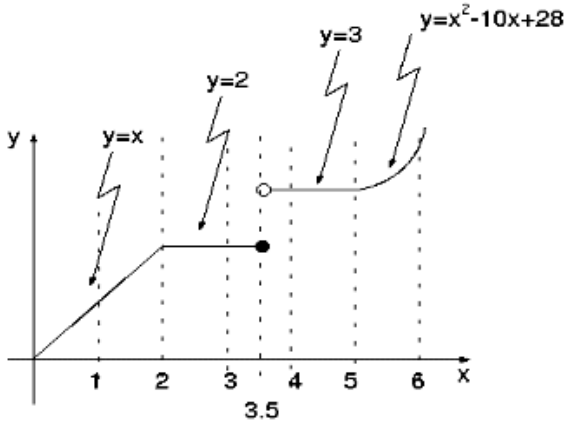
```
'''Função que recebe como entrada um texto e o número n de repetições desejado e retorna uma sequê
'''str, int -> str'''
# 1 caso de teste: um texto qualquer. Prestar atenção nos espaços de separação entre as repetições

def repeteTexto(texto, n):
    return n*texto
```

4. Defina uma função em Python que receba como entrada três números inteiros representando, respectivamente, dia, mês e ano. Sua função deve retornar uma sequência de caracteres com estas informações formatadas no padrão usual de notação de datas: "DD/MM/AAAA"

Solution:

```
def formataData(d, m, a):
    '''retorna uma data formatada int, int, int -> str'''
    return str(d)+"/"+str(m)+"/"+str(a)
```

5. Defina uma função em Python que tenha o comportamento da função matemática da figura abaixo para valores de entrada maiores ou iguais a zero. Para valores negativos o valor da função é sempre zero:



Qual o número mínimo de casos de teste para garantir que todas as linhas do programa são executadas? Crie casos de teste para os pontos de inflexão da função (as fronteiras entre cada parte).

Solution:

```
# Função que dado o valor de x retorna o valor y de uma função matemática
# float -> float ou str(caso x<0)
# 5 casos de teste: um para cada intervalo da função e mais x<0,
# para o qual a função não está definida

def funcao(x):
```

```
    if x<0:
        return 0
    elif x<=2:
        return x
    elif x<=3.5:
        return 2
    elif x<5:
        return 3
    return x**2 - 10*x+28
```

6. Sabe-se que, pelas regras trabalhistas, percentuais de desconto de INSS (Previdência Social) e IR (Imposto de Renda) variam de acordo com a faixa salarial de um funcionário. O cálculo do salário líquido a partir do salário bruto é dado pela expressão

$$salariolíquido = salariobruto - \sum descontos.$$

Dado como parâmetro de entrada o valor do salário bruto, construa as funções a seguir, tendo o cuidado de informar nas descrições qualquer informação que você considere importante para o usuário da função.

(a) uma função que calcule e retorne o valor do desconto de imposto de INSS de acordo com a tabela do INSS, ou seja,
6% para salário bruto até R$2.000,00 (inclusive);
8% para salário bruto até R$3.000,00 (inclusive);
10% para salário bruto acima de R$3.000,00;

(b) uma função que calcule e retorne o valor do desconto de IR de acordo com a tabela do IR, ou seja,
11% para salário bruto até R$2.500,00 (inclusive);
15% para salário bruto até R$5.000,00 (inclusive);
22% para salário bruto acima de R$5.000,00;

(c) uma função que calcule e retorne o salário líquido, usando as funções criadas nos itens a e b.

Solution:

```
def calcula_desconto_IR (bruto):
    ''' calcula a taxa de desconto do IR dado o salário bruto.'''
    ''' float --> float '''
    if (bruto <= 2500.00):
        return 0.11*bruto
    elif (bruto <=5000.00):
        return 0.15*bruto
    else:
        return 0.22*brutp


def calcula_desconto_Inss(bruto):
    ''' calcula a taxa de desconto do INSS dado o salário bruto.'''
    ''' float --> float '''
    if (bruto <= 2000.00):
        return 0.06*bruto
    elif (bruto <= 3000.00):
```

```
        return 0.08*bruto
    else:
        return 0.10*bruto

def salarioLiquido (bruto):
    ''' calcula salário líquido, descontados o IR e o INSS, dado o salário bruto.'''
    ''' float --> float '''
    return bruto - (calcula_desconto_IR (bruto) + calcula_desconto_Inss(bruto))
```