Mundo 3 - Funções simples

1) Faça um programa que tenha uma função chamada area(), que receba as dimensões de um terreno retangular (largura e comprimento) e mostre a área do terreno

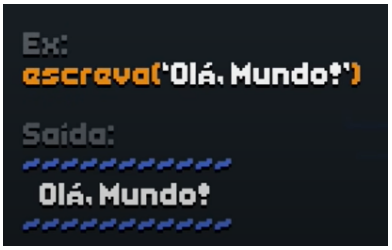
a)

```
def calculo_area(a,b):
  area = a*b
  print(f'Area = {area}m²')

comp = int(input('Comprimento: '))
larg = int(input('Largura: '))
calculo_area(comp, larg)
```

Guanabara)

```
def area(larg,comp):
    a = larg * comp
    print(f'A area de um terreno {larg}x{comp} é de {a}m²')


print('Controle de terrenos')
print('-'*20)
l = float(input('LARGURA (m): '))
c = float(input('COMPRIMENTO (m): '))
area(l,c)
```

2) Faça um programa que tenha uma função chamada escreva(), que receba um texto qualquer como parâmetro e mostre uma mensagem com tamanho adaptável



a)

Desta forma nós escrevemos apenas uma mensagem com caracteres cobrindo o texto, mas e se quisermos que os caracteres vão apenas até o tamanho do texto?

```
def escreva(x):
  print('{}{}{}'.format('\033[32m','~'*30,'\033[m'))
  print(x)
  print('{}{}{}'.format('\033[32m','~'*30,'\033[m'))

escreva(input('Digite algo: '))
```

Guanabara)

A variável tam recebe a quantidade da len da mensagem então quando for printado os elementos será mostrado apenas o tamanho da própria msg

```
def escreva(msg):
    tam = len(msg)
    print('~'*tam)
    print(msg)
    print('~'*tam)

escreva('Gustavo Guanabara')
```

Se quiséssemos por exemplo colocar borda e centralizar a mensagem:

```
def escreva(msg):
    tam = len(msg)+4
    print('~'*tam)
    print(f'  {msg}')
    print('~'*tam)

escreva('Gustavo Guanabara')
```

ou fazemos uma configuração diferente de f string

```
def escreva(msg):
    tam = len(msg)+4
    print('~'*tam)
    print('{:^{}}'.format(msg,tam))
    print('~'*tam)


escreva('Gustavo Guanabara')
```

3) Faça um programa que tenha uma função chamada contador() que receba três parâmetros: inicio, fim e passo e realize a contagem. Seu programa tem

que realizar três contagens através da função criada

- de 1 até 10 de 1 em 1
- de 10 até 0 de 2 em 2
- uma contagem personalizada

a)

Neste exemplo, passamos parâmetros de forma direta e o programa já faz o calculo

```
from time import sleep
def msg():
    print('-'*50)
def contador(i,f,p):
    msg()
    print('Contando de 1 a 10 ( de 1 em 1 )')
    for c in range(1,11):
        sleep(0.4)
        print(c, end='-> ')
    print('FIM')
    msg()
    print('Contando de 10 a 0 ( de 2 em 2 )')
    for c in range(10,-1,-1):
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg()
    print(f'Contando de {i} a {f} ( de {p} em {p} )')
    for c in range(i,f+1,p):
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg


contador(0,200,20)
```

b)

Podemos criar a função sem parâmetros e criar dentro da mesma:

```
from time import sleep
def msg():
    print('-'*50)
def contador():
    i = int(input('Inicio: '))
    f = int(input('Fim: '))
```

```
    p = int(input('Passo: '))
    msg()
    print('Contando de 1 a 10 ( de 1 em 1 )')
    for c in range(1,11):
        sleep(0.4)
        print(c, end='-> ')
    print('FIM')
    msg()
    print('Contando de 10 a 0 ( de 2 em 2 )')
    for c in range(10,-1,-1):
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg()
    print(f'Contando de {i} a {f} ( de {p} em {p} )')
    for c in range(i,f+1,p):
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg


contador()
```

c)

Podemos criar as variáveis na chamada da def

```
from time import sleep
def msg():
    print('-'*50)
def contador(i,f,p):
    msg()
    print('Contando de 1 a 10 ( de 1 em 1 )')
    for c in range(1,11):
        sleep(0.4)
        print(c, end='-> ')
    print('FIM')
    msg()
    print('Contando de 10 a 0 ( de 2 em 2 )')
    for c in range(10,-1,-1):
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg()
    print(f'Contando de {i} a {f} ( de {p} em {p} )')
    for c in range(i,f+1,p):
```

```
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg


contador(i = int(input('Inicio: ')),
    f = int(input('Fim: ')),
    p = int(input('Passo: ')))
```

d)

Podemos passar como posição de tupla ou lista

```
from time import sleep
def msg():
    print('-'*50)
def contador(valores):
    msg()
    print('Contando de 1 a 10 ( de 1 em 1 )')
    for c in range(1,11):
        sleep(0.4)
        print(c, end='-> ')
    print('FIM')
    msg()
    print('Contando de 10 a 0 ( de 2 em 2 )')
    for c in range(10,-1,-1):
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg()
    print(f'Contando de {valores[0]} a {valores[1]} ( de {valores[2]} em {v
alores[2]} )')
    for c in range(valores[0],valores[1]+1,valores[2]):
        sleep(0.4)
        print(c,end='-> ')
    print('FIM')
    msg


numeros = (int(input('Inicio: ')),
    int(input('Fim: ')),
    int(input('Passo: ')))

contador(numeros)
```

Guanabara)

O Guanabara resolveu de uma forma um pouco diferente, ele usa a estrutura While, mostrando o fim exclusivo, onde podemos usar o sinal <= ou >=.

Então entendendo este programa passo a passo seria:

1. importamos a função sleep da biblioteca time
2. criamos a def chamada contador passando tres parâmetros
    a. Se o p for menor que 0 então p recebe ele mesmo vezes -1 para ficar positivo (utilizamos esta função pois se o usuário quiser contar de 10 a 0 o que conta é o contador e ele vai diminuindo até que seja igual ao fim, como ele faz c = c - p e se o p fosse positivo, ficaria c = c - (- p), logo aumentaria e ficaria rodando infinitamente
    b. Se o p for zero, não existe p 0, senão ele printa o i infinitamente também então no mínimo p vale 1
3. printamos uma linha para separar o código
4. printamos algumas informações de contagem
5. damos um sleep de 0.5 segundos
6. Agora entra a parte de contar com o while, se o i for menor que o f:
    a. declaramos cont como o i
    b. enquanto o cont for menor ou igual a f
    c. printa o cont na mesma linha
    d. sleep de 0.5 segundos
    e. cont recebe ele mais p a cada loop
    f. printamos fim
7. Senão:
    a. declaramos cont como i
    b. Enquanto cont for maior ou igual a f
    c. printamos cont na mesma linha
    d. damos sleep de 0.5
    e. cont recebe cont + p
    f. printamos fim
8. FInalizando o programa principal, pois até então só criamos a def
9. função contador recebe parâmetros i = 0, f = 100, p = 10
10. função contador recebe parâmetros i = 10,f = 0, p= 2
11. printamos agora um espaço para o usuário passar suas informações, sendo o i, f e p que estarão sendo enviados à def contador
12. E na última linha temos o contador puxando as informações das variáveis globais

```
from time import sleep

def contador(i,f,p):
    if p < 0:
        p *= -1
    if p == 0:
        p = 1
    print('-='*20)
    print(f'Contagem de {i} até {f} de {p} a {p}')
    sleep(0.5)
    if i < f:
        cont = i
        while cont <= f:
            print(f'{cont} ',end = '')
            sleep(0.5)
            cont += p
        print('Fim!')
    else:
        cont = i
        while cont >= f:
            print(f'{cont} ',end='')
            sleep(0.5)
            cont-=p
        print('Fim!')
    sleep(0.5)
    #Programa Principal
contador(0,100,10)
contador(10,0,2)
print('-='*30)
print('Agora é a sua vez de personalizar a contagem!')
ini = int(input('Inicio: '))
fim = int(input('Fim: '))
pas = int(input('Passo: '))
contador(ini,fim,pas)
```

4) Faça um programa que tenha uma função chamada maior() que receba vários parâmetros com valores inteiros. Seu programa tem que analisar todos os valores e dizer qual deles é maior.

a)

Vou tentar fazer o mesmo exercicio, mas com while

```
from time import sleep
def maior(*num):
    cont = maior = 0
```

```
    print('-='*30)
    print('Analisando os valores passados...')
    while cont < len(num):
        print(f'{num[cont]} ',end='')
        sleep(0.3)
        if cont == 0:
            maior = num[cont]
        else:
            if num[cont] > maior:
                maior = num[cont]
        cont+=1
    print(f'Foram informados {cont} valores ao todo')
    print(f'O maior valor informado foi {maior}')


    #Programa principal
maior(2,9,4,5,7,1)
maior(4,7,0)
maior(1,2)
maior(6)
maior(0)
```

b)

Tem como passar também como lista, podendo usar já uma função built in que é o max( )

```
from time import sleep
def maior(lst):
    cont = 0
    maior = max(lst)
    print('-='*30)
    print('Analisando os valores passados...')
    while cont < len(lst):
        print(f'{lst[cont]} ',end='')
        sleep(0.3)
        cont+=1
    print(f'Foram informados {cont} valores ao todo')
    print(f'O maior valor informado foi {maior}')


    #Programa principal
lista = []
while True:
    lista.append(int(input('Digite um número: ')))
    resp = input('Deseja continuar [S/N] ?: ').upper()[0]
    while True:
        if resp in 'SN':
```

```
            break
        print('ERRO ! Favor digitar apenas S ou N')
    if resp == 'N':
        break
maior(lista)
```

Guanabara)

Vamos enteneder o que o código faz:

1. primeiramente importamos a função sleep da biblioteca time
2. criamos a def maior recebendo o parâmetro *num onde puxará todos os valores dentro da tupla
3. variáveis cont e maior recebem o valor 0
4. printamos -= * 30
5. printamos “Analisando os valores passados...”
6. for valor em num (ou seja vamos iterar em cima de num com a variável valor, recebendo cada posição em cada loop
    a. printamos o valor juntando as linhas
    b. damos um sleep de 0.3
    c. if o cont for 0 então maior recebe o valor em num
    d. senão, se o valor for maior que maior então maior recebe valor
    e. cont recebe ele mais 1
7. Foram informados cont valores ao todo
8. O maior número informado foi maior
9. Para o programa principal, passamos o maior com vários números diferentes dentro dele.

```
from time import sleep

def maior(*num):
    cont = maior = 0
    print('-='*30)
    print('Analisando os valores passados...')
    for valor in num:
        print(f'{valor} ',end='')
        sleep(0.3)
        if cont == 0:
            maior == valor
        else:
            if valor > maior:
                maior == valor
        cont += 1
```

```
    print(f'Foram informados {cont} valores ao todo')
    print(f'O maior valor informado foi {maior}')

    #Programa principal
maior(2,9,4,5,7,1)
maior(4,7,0)
maior(1,2)
maior(6)
maior(0)
```

5) Faça um programa que tenha uma lista chamada números e duas funções chamadas sorteia() e somaPar(). A primeira função vai sortear 5 números e vai colocá-los dentro da lista e a segunda função vai mostrar a soma entre todos os valores pares sorteados pela função anterior.

a)

```
from random import randint
from time import sleep

def sorteia(numeros):
    print('Sorteando 5 valores da lista: ',end='')
    for c in range(5):
        numeros.append(randint(0,10))
    for i, c in enumerate(numeros):
        print(numeros[i],end=' ')
        sleep(0.4)

def somaPar(numeros):
    soma = 0
    for i, c in enumerate(numeros):
        if numeros[i] % 2 == 0:
            soma += numeros[i]
    print(f'\nSomando os valores pares de {numeros}, temos {soma}')


#   Programa Principal
numeros = list()
sorteia(numeros)
somaPar(numeros)
```

Guanabara)

Tivemos algumas diferenças entre o meu código e o do Guanabara, sendo elas:

- Na primeira função, não ter a necessidade de varrer o código, basta inserirmos o randint em uma variável, dar append dela na lista e printa ela a cada sleep
- Na segunda função, não precisamos o enumerate, podemos fazer pelo for normal na lista e iterar em cima do for para somar.

```
from random import randint
from time import sleep

def sorteia(lista):
    print('Sortenado 5 valores da lista: ',end='')
    for cont in range(0,5):
        n = randint(1,10)
        lista.append(n)
        print(f'{n} ',end='')
        sleep(0.3)
    print('PRONTO!')

def somaPar(lista):
    soma = 0
    for valor in lista:
        if valor % 2 == 0:
            soma += valor
    print(f'Somando os valores pares de lista {lista}, temos {soma}')

numeros = list()
sorteia(numeros)
somaPar(numeros)
```