**Programas da Aula 07:**

# Programas da aula 07

def mensagemDeErro():

    """Não retornar nada, mas mostra na tela uma mensagem de erro clássica"""

    print('Tudo o que foi escrito está errado. \n')

    print('Por favor, comece tudo do zero.')

mensagemDeErro()

def add(x, y):

    """Retorne x mais y"""

    return x + y

primeiroParamentro = int(input("Digite o primeiro número"))

SegundoParamentro = int(input("Digite o segundo número"))

print(add(primeiroParamentro, SegundoParamentro))

def arbitrary(x, y, \*more):

    """Criando uma função que recebe mais parâmetros"""

    print("X=",x,", y=",y)

    print("arbritary: ",more)

arbitrary(3,4)

arbitrary(3,4, "Hello word", 3, 4)

def func():

    x = 10

    print("Valor dentro da função: ", x)

x = 20

func()

print("Valor fora da função:", x)

def func(country = "Norway"):

    print('I am from '+country)

func("Sweden")

func('Brazil')

func()

func("Japan")

def func(food):

    for x in food:

        print(x)

frutas = ["Maçã", "Banana", "Morango"]

func(frutas)

def recursao(k):

    if(k>0):

        result = k+recursao(k-1)

        print(result)

    else:

        result = 0

    return result

print("\n\nRecursao exemplos")

recursao(20)

cube = lambda x: x\*x\*x

print(cube(7))

def somaDeListas(lista1,lista2):

    """Essa função soma duas listas"""

    for elementos in zip(lista1,range(0,len(lista1))):

        lista2[elementos[1]] = lista2[elementos[1]] + elementos[0]

    return lista2

lista1 = [1, 2, 3, 4, 5]

lista2 = [1, 2, 3, 4, 5]

print(somaDeListas(lista1,lista2))

print(somaDeListas.\_\_doc\_\_)

def lerlista():

    lista = [0,0,0,0,0,0,0,0,0,0]

    for i in range(0,10):

        lista[i] = float(input('Digite 10 elementos para lista \n'))

    return lista

print(lerlista())

class Point:

    def \_\_init\_\_(self,initX,initY):

        self.x = initX

        self.y = initY

    def getPointValue(self):

        return [self.x, self.y]

    def hipo(self):

        return (self.x\*\*2 + self.y\*\*2)\*\*0.5

    def half(self, target):

        return (self.x+target.x)/2 + (self.y+target.y)/2

a = Point(34,85)

b = Point(3,9)

print(a.getPointValue())

print(a.hipo())

print(a.half(b))

class Point:

        def \_\_init\_\_(self, initX, initY):

            self.x = initX

            self.y = initY

        def getX(self):

            return self.x

        def getY(self):

            return self.y

        def distanceFromOrigin(self):

            return ((self.x \*\* 2)+ (self.y \*\* 2)) \*\* 0.5

        def \_\_str\_\_(self):

            return "x=" + str(self.x) + ", y=" + str(self.y)

        def halfway(self, target):

            mx = (self.x + target.x)/2

            my = (self.y + target.y)/2

            return Point(mx, my)

p = Point(3,4)

q = Point(5,12)

mid = p.halfway(q)

print(mid)

print(mid.getX())

print(mid.getY())

print(mid.distanceFromOrigin())

**Atualização IMC:**

# Aula 07 - Atualização do programa de IMC para o 4.0

def imc(altura, peso):

    """Retornar IMC"""

    return peso / (altura\*\*2)

print('Índice de Massa Corpotal (IMC)')

altura = float(input('Por favor, digite a sua altura:'))

peso = float(input('Por favor, digite o seu peso:'))

print("Seu IMC é: ", imc(altura,peso))

print('Você está:')

if imc(altura,peso) < 17:

    print('Muito abaixo do peso.')

if 17 <= imc(altura,peso) < 18.5:

    print('Abaixo do peso normal.')

if 18.5 <= imc(altura,peso) <25:

    print('Dentro do peso normal.')

if 25 <= imc(altura,peso) < 30:

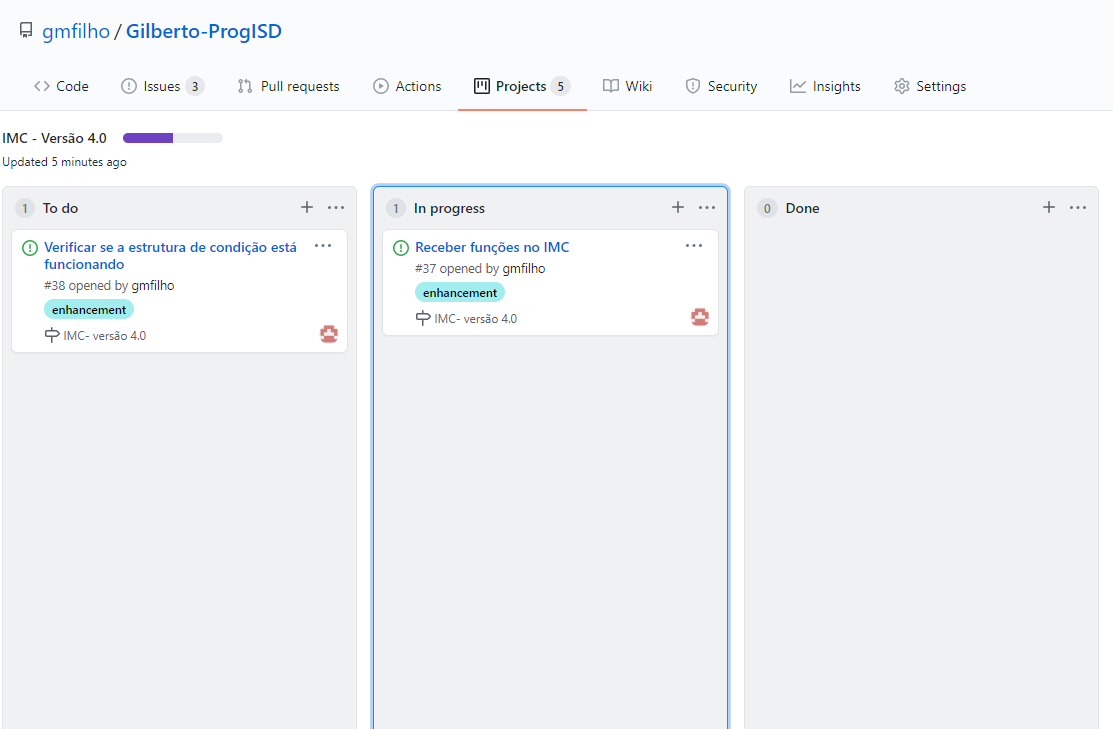
    print('Acima do peso normal.')

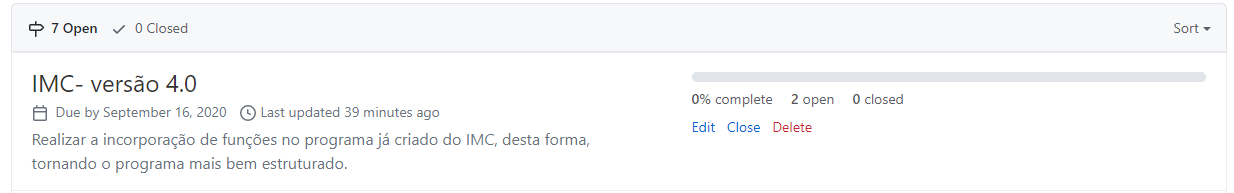
if imc(altura,peso)>30:

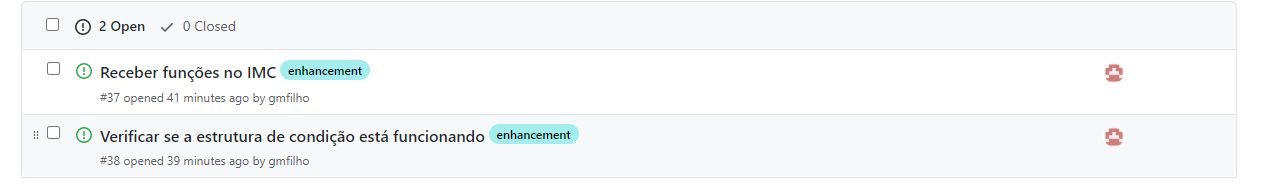
    print('Muito acima do peso.')

print('')

print('Cuide da sua saúde e faça exercícios físicos.')







**Atualizando Caixa Comportamental:**

#Exercício da Aula 07 - Atualização da caixa comportamental v3.0

def treino():

    for i in range(50):

        som1 = str(input('Som 1 foi tocado e o animal tocou a barra da esquerda? "s" para SIM e "n" para NÃO.\n'))

        som2 = str(input('Som 2 foi tocado e o animal tocou a barra da direita? "s" para SIM e "n" para NÃO.\n'))

        if (som1 == "s" and som2 == "s"):

            print('Som 1: Liberar 0,5ml de rec.')

            print('Som 2: Liberar 0,5ml de rec.')

        elif (som1 == "n" and som2 =="s"):

            print('Som 1: Não liberar nada.')

            print('Som 2: Liberar 0,5ml de rec.')

        elif (som1 == "s" and som2 == "n"):

            print('Som 1: Liberar 0,5ml de rec.')

            print('Som 2: Não liberar nada.')

        else:

            print('Não liberar nada')

    return print("Fim do treinamento")

def habituar():

    distancia=30

    for i in range(20):

        distancia1=float(input("Qual a distância que o sagui está da barra?\n"))

        if (distancia1 < distancia):

            print("Liberar 0,5ml de rec.")

        else:

            print("Não liberar nada")

    return print('\nFim da habituação do animal. \nVamos para a próxima fase!')

def temp():

    tempo=float(input('Por favor, digite o tempo utilizado no treinamento em minutos:\n'))

    if (tempo > 30):

        print('O experimento excedeu os 30 minutos, logo não será considerado. Por favor, realize novamente o experimento após o animal descansar no mínimo 4 horas.')

    else:

        print('O expermento foi realizado com sucesso e seguirá para a próxima fase.')

    return print('Fim do treinamento.')

def saguis():

    print('<<Programa para treinamento de saguis>>')

    print('')

    print('1ª Etapa: Habituação do animal na caixa.')

    habituado = str(input('O animal está habituado? "s" para SIM e "n" para NÃO.\n'))

    if (habituado == "s"):

        print("Vamos iniciar o treinamento!")

        print('')

        habituar()

        treino()

        temp()

    else:

        print('Abra novamente o programa quando o animal estiver melhor habituado.')

saguis()

