Auxiliar semana 2

P1) En el modulo triangulo.py se encuentran las siguientes funciones:

- **esTriangulo**: Entrega un booleano verificando si 3 lados pueden formar un triángulo
- **perimetroTriangulo**: Entrega el valor del perímetro de un triángulo dado sus 3 lados
- **areaTriangulo**: Entrega el valor del área del triángulo dados sus 3 lados

Programe la función **datosTriangulo** usando el modulo triangulo.py donde dado 3 valores verifica si es un triangulo, si es así muestra en pantalla el valor del perímetro y área, en caso contrario imprime en pantalla, no es un triángulo. Siga el siguiente esquema:

```
datosTriangulo(2,3,4)
perimetro: 9
area: 2.9047375096555625



datosTriangulo(8,1,2)
no es un triangulo
```

```
# Importamos el modulo triangulo
import triangulo


# triangulo int int int -> bool
# Entrega el perimetro y area de un triangulo si los 3
# Lados pueden formar un triangulo
# Ejemplo: datosTriangulo(3,3,5)
def datosTriangulo(lado1,lado2,lado3):

  # Verificamos que sea un triangulo
  if triangulo.esTriangulo(lado1,lado2,lado3):

    # Calculamos su perimetro y area, entregando los dos
    perimetro = triangulo.perimetroTriangulo(lado1,lado2,lado3)
    area = triangulo.areaTriangulo(lado1,lado2,lado3)
    print("perimetro: " + str(perimetro))
    print("area: " + str(area))
    return True

  # Caso donde no es un triangulo
  else:

    print("no es un triangulo")
```

```
    print( 'no es un triangulo' )
    return False

# Test
assert datosTriangulo(3,3,5)
```

```
perimetro: 11
area: 4.14578098794425
```

P2) Queremos realizar un programa que verifique si un numero a es **diminutivo** de otro numero b otro, es decir , si cada uno de los dígitos de a es menor estricto a cada uno de los dígitos de b

Ejemplo: 523 es diminutivo de 687. En cambio 418 no es diminutivo a 293

Cree la función diminutivo(a,b), en donde a y b son números enteros, cada uno de 3 cifras. Para esto siga el siguiente comportamiento:

- Si a es diminutivo de b, entonces la función debe mostrar "correcto" en pantalla retornando un numero 1.
- En caso contrario, debe mostrar "incorrecto! n_a es mayor o igual a n_b", donde n_a y n_b son los dígitos donde no se cumple la condición y finalmente retornar un numero 0

```
# diminutivo: int, int -> int
# Imprime en pantalla si a es diminutivo de b, en caso de no serlo indica cual digito
# No cumple con la condicion
# Ejemplo: diminutivo(123,456) debe imprimir correcto! y retornar un 1

def diminutivo(a, b):

  # Sacamos cada digito de los numeros
  unidad_a = a % 10
  decena_a =(a//10)%10
  centena_a =( a//100)%10
  unidad_b = b % 10
  decena_b =(b//10)%10
  centena_b =(b//100)%10

  # realizamos las comparaciones para unidad
  if unidad_a >= unidad_b:
    print ('incorrecto!', unidad_a, 'es mayor o igual que', unidad_b)
    return 0

   # realizamos las comparaciones para decena
  elif decena_a >= decena_b:
    print ('incorrecto!' + str(decena_a) + 'es mayor o igual que' + str(decena_b))
    return 0 # finalmente comparamos la centena

  elif centena_a >= centena_b:
    print ('incorrecto!' + str(centena_a) + 'es mayor o igual que' + str(centena_b))
```

```
    return 0

  # Cumple la condicion de ser diminutivo
  else:
    print ('correcto!')
    return 1

# Test
assert diminutivo(123, 333) == 0
assert diminutivo(123, 456) == 1
assert diminutivo(745, 689) == 0
assert diminutivo(444, 222) == 0
```

P3) En plena cuarentena te encuentras en tu casa a punto de entregar una tarea para el ramo de programación, pero ha ocurrido un problema, te has quedado sin internet, a lo que tienes que elegir una de las siguientes alternativas:

- Correr donde tu amigo más cercano (amigo_x,amigo_y) para pedirle internet y enviar tu tarea
- Ir donde tu abuela (abuela_x,abuela_y) para conseguir internet y lograr enviar tu tarea

Para esto necesitas calcular el lugar más cercano y así tomar una buena decisión. Además cuentas con un permiso de 30 minutos (si no iras detenido), consideramos que la velocidad sera de 15km/h en bicicleta (trabaje en minutos, las distancias están en km). Para lograrlo siga los siguientes pasos:

A)

- i) **distancia(x1,x2,y1,y2)**: Esta calcula la distancia entre dos puntos
- ii) **tiempo(distancia)**: Esta debe calcular el tiempo que demora en recorrer esa distancia

```
import math


# distancia: float float float float -> float
# Calcula la distancia entre 2 puntos
# Ejemplo: distancia(0,0,0,5) debe entregar un 5
def distancia(x1,y1,x2,y2):
  return math.sqrt((x1-x2)**2 + (y1-y2)**2)

assert distancia(0,0,0,5) == 5.0
assert distancia(5,4,5,1) == 3.0

# tiempo: float -> float
# Calcula la cantidad de minutos que demora en recorrer cierta distancia
# Ejemplo: tiempo(5.0) entrega 20
def tiempo(distancia):

  # Pasamos km/h a km/minutos
```

```
  velocidad = 15/60
  return distancia/velocidad

assert tiempo(5.0) == 20
```

```
20.0
```

B) Realice un programa interactivo que solicite su ubicación, la ubicación de su amigo y de su abuela, finalmente entrege como resultado la ubicación más cercana y si es posible llegar a tiempo sin que el permiso expire. Para esto siga la siguiente plantilla:

```
Su ubicacion x? 5
Su ubicacion y? 6
Amigo x? 5
Amigo y? 5
Abuela x? 7
Abuela y? 9
Menor distancia: amigo
Tiempo:3.75
Lo logra
```

```
# Pedimos las ubicaciones
ubicacion_x = input("Su ubicacion x? ")
ubicacion_y = input("Su ubicacion y? ")
amigo_x = input("Amigo x? ")
amigo_y = input("Amigo y? ")
abuela_x = input("Abuela x? ")
abuela_y = input("Abuela y? ")

# Calculamos cada distancia por separado
distancia_amigo = distancia(int(ubicacion_x), int(ubicacion_y), int(amigo_x), int(amigo_y))
distancia_abuela = distancia(int(ubicacion_x), int(ubicacion_y), int(abuela_x), int(abuela_y)

# Si el amigo se encuentra mas cerca
if distancia_amigo <= distancia_abuela:

  # Calculamos el tiempo que demora en ir donde el amigo
  tiempo_amigo = tiempo(distancia_amigo)
  print("Menor distancia: amigo")
  print("Tiempo:" + str(tiempo_amigo))
  if tiempo_amigo <= 30:
    print("Lo logra")
  else:
    print("No lo logra")
```

```
 # Si la abuela se encuentra mas cerca
else:

  # Calculamos el tiempo que demora en ir donde la abuela
  tiempo_abuela = tiempo(distancia_abuela)
  print("Menor distancia: abuela")
  print("Tiempo:" + str(tiempo_abuela))
  if tiempo_abuela <= 30:
    print("Lo logra")
  else:
    print("No lo logra")
```

P4) Para esta pregunta vamos a recrear la caracola mágica de Bob Esponja pero con algunas modificaciones. Nuestra caracola tendrá tres respuestas para nuestras preguntas, estas son "Si!", "No!" o "No he entendido la pregunta". Para esto, programe una función caracola(pregunta) la cual entregara la respuesta de la siguiente manera:

- El 45% de las veces responderá con un "Si!"
- El 35% de las veces responderá con un "No!"
- El resto del 20% responderá con un "No lo he entendido"

Además de esta función, realice un programa interactivo para poner a prueba la caracola, siga el siguiente ejemplo:

```
Cual es la pregunta: ¿Pasare este ramo?
Respuesta: Si!
```

```
import random # Importamos el modulo random!!

# caracola: str -> str
# entrega una respuesta con un Si!, No! o No lo he entendido a las preguntas que
# se le entregan
# Ejemplo: caracola("¿Pasare el ramo?") el 45% de la veces entrega un Si! como respuesta
def caracola(pregunta):
  # la idea es tomar un numero al azar del 1 al 100 y ver entre cuales numeros está
  n=random.randint(1,100)

  # 45% responde un Si!
  if n>=1 and n<=45:
    return "Si!"

  # 35% responde un No!
  if n>45 and n<=75:
    return "No!"
```

```
    return "No!"

  # El resto del 20% responde No lo he entendido
  else:
    return "No lo he entendido"

#Test
test = caracola("¿Soy cool?")
assert test == "Si!" or test == "No!" or test == "No lo he entendido"
```

```
pregunta = input("Cual es la pregunta:")
respuesta = caracola(pregunta)
print("Respuesta: " + respuesta )
```