****

**Desafío entregable 5 (Clase 9 y 10)**

**"Funciones"**

**1) Realiza una función llamada area\_rectangulo() que devuelva el área del rectángulo a partir de una base y una altura. Calcula el área de un rectángulo de 15 de base y 10 de altura**

**🖐 Ayuda:** El área de un rectángulo se obtiene al multiplicar la base por la altura.

|  |  |
| --- | --- |
| Respuesta | rd={'base':int,'units':'cm = centimeters',      'height':int,'units':'cm = centimeters',      'docstring':'If you want to calculate the area of ​​a space, do it by multiplying its length by its width. So you will get a certain amount of square meters. This is the standard procedure for calculating rectangles. For example: a room measures 30 meters by 15 meters.'}  def area\_rectangulo(base:'int',height:'int')->rd:      try:          base = int(base)          height = int(height)          if type(base)==int and type(height)==int:              return base \* height          else:              return False      except ValueError:          print('Error, el parametro de base : {} o de altura {} no son numeros enteros, debe cambiarlos para continuar '.format(base,height))          return False    if \_\_name\_\_ == '\_\_main\_\_':      assert area\_rectangulo(15,10) == 150, 'ok'      assert area\_rectangulo(15.0,10.0) == 150 , 'ok'      print("\*" \* 100)      print( area\_rectangulo.\_\_annotations\_\_['return'] )      print(  area\_rectangulo(15,10)  )      base = input(f"Ingrese valor de la base :")      altura = input(f"Ingrese valor de la altura :")      print(  area\_rectangulo(base,altura)  ) |

**2) Realiza una función llamada area\_circulo() que devuelva el área de un círculo a partir de un radio. Calcula el área de un círculo de 5 de radio**

**🖐 Ayuda:** El área de un círculo se obtiene al elevar el radio a dos y multiplicando el resultado por el número pi. Puedes utilizar el valor 3.14159 como pi o importarlo del módulo math.

|  |  |
| --- | --- |
| Respuesta | import math  rd={'radio':int,'units':'cm = centimeters',      'docstring':'The area of ​​a circle is pi multiplied by the radius squared (A = π r²).'}  def area\_circulo(radio:'int')->rd:      try:          radio = int(radio)          if type(radio)==int :              return   round((math.pi \* (radio \*\* 2) ),4)          else:              return False      except ValueError:          print('Error, el parametro de radio : {}  no son numeros enteros, debe cambiarlos para continuar '.format(radio))          return False    if \_\_name\_\_ == '\_\_main\_\_':      assert area\_circulo(5) == 78.5398, 'ok'      assert area\_circulo(5.0) == 78.5398 , 'ok'      print("\*" \* 100)      print( area\_circulo.\_\_annotations\_\_['return'] )      print(  area\_circulo(5)  )      radio = input(f"Ingrese valor del radio :")      print(  area\_circulo(radio)  ) |

**3) Realiza una función llamada relacion() que a partir de dos números cumpla lo siguiente:**

1. **Si el primer número es mayor que el segundo, debe devolver 1.**
2. **Si el primer número es menor que el segundo, debe devolver -1.**
3. **Si ambos números son iguales, debe devolver un 0.**

**Comprueba la relación entre los números: '5 y 10', '10 y 5' y '5 y 5'**

|  |  |
| --- | --- |
| Respuesta | rd={'number01':int,'number':'integers',      'number02':int,'number':'integers',      'docstring':'relationship between 2 numbers to know if the first number is greater than, less than or equal to the second number.'}  def relacion(number01:'int',number02:'int')->rd:      try:          number01 = int(number01)          number02 = int(number02)          if type(number01)==int and type(number02)==int :              if number01 > number02 :                  return 1              elif number01 < number02 :                  return -1              else :                  return 0          else:              #assert False,"parameter numero #01 o numero #02 no son numero enteros"              return False      except ValueError:          print('Error, el parametro de numero #1 : {} y parametro numero #2 : {} no son numeros enteros, debe cambiarlos para continuar '.format(number01,number02))          # assert False,"Error, el parametro de numero #1 : {} o el parametro numero #2 : {} no son numeros enteros, debe cambiarlos para continuar ".format(number01,number02)          return False    if \_\_name\_\_ == '\_\_main\_\_':      assert relacion(5,10) == -1, 'ok'      assert relacion(10,5) == 1, 'ok'      assert relacion(5,5)  == 0, 'ok'      assert relacion(5.0,'a')  == False, 'ok'          print("\*" \* 100)      print( relacion.\_\_annotations\_\_['return'] )      print( relacion(5,10)  )      print( relacion(10,5)  )      print( relacion(5,5)  )      numero1 = input(f"Ingrese valor del numero #1 :")      numero2 = input(f"Ingrese valor del numero #2 :")      print(  relacion(numero1,numero2)  ) |

**4)Realiza una función llamada intermedio() que a partir de dos números, devuelva su punto intermedio:**

**🖐 Ayuda:** El número intermedio de dos números corresponde a la suma de los dos números dividida entre 2

**Comprueba el punto intermedio entre -12 y 24**

|  |  |
| --- | --- |
| Respuesta | rd={'number01':int,'number':'integers',      'number02':int,'number':'integers',      'docstring':'The midpoint between two points is a point that has coordinates that lie exactly midway between the two points. These coordinates can be found by adding the x-coordinates of the two points and dividing by 2.'}  def intermedio(number01:'int',number02:'int')->rd:      try:          number01 = int(number01)          number02 = int(number02)          if type(number01)==int and type(number02)==int :              # assert True,"Calculo de punto medio satisfactorio"              return ( number01 + number02 ) / 2          else:              # assert False,"parameter numero #01 o numero #02 no son numero enteros"              return False      except ValueError:          print('Error, el parametro de numero #1 : {} y parametro numero #2 : {} no son numeros enteros, debe cambiarlos para continuar '.format(number01,number02))          # assert False,"Error, el parametro de numero #1 : {} o el parametro numero #2 : {} no son numeros enteros, debe cambiarlos para continuar ".format(number01,number02)          return False    if \_\_name\_\_ == '\_\_main\_\_':      #print ( intermedio(-12,24) )      assert intermedio(-12,24)  == 6, 'Prueba Satisfactoria parametros Numer01 :-12, Numer02 : 24'          # print("\*" \* 100)      # print( intermedio.\_\_annotations\_\_['return'] )      # print( intermedio(5,10)  )      # print( intermedio(10,5)  )      # print( intermedio(5,5)  )      print ( intermedio(-12,24) )      numero1 = input(f"Ingrese valor del numero #1 :")      numero2 = input(f"Ingrese valor del numero #2 :")      print(  intermedio(numero1,numero2)  ) |

**5) Realizá una función llamada recortar() que reciba tres parámetros. El primero es el número a recortar, el segundo es el límite inferior y el tercero el límite superior. La función tendrá que cumplir lo siguiente:**

1. **Devolver el límite inferior si el número es menor que éste**
2. **Devolver el límite superior si el número es mayor que éste.**
3. **Devolver el número sin cambios si no se supera ningún límite.**

**Comprueba el resultado de recortar 15 entre los límites 0 y 10**

|  |  |
| --- | --- |
| Respuesta | rd={'numberToCut':int,'number':'integers',      'lowerLimit':int,'number':'integers',      'upperimit':int,'number':'integers',      'docstring':'UPPER LIMIT AND LOWER LIMIT In mathematics, the upper limit and lower limit of a sequence are defined as the highest and lowest limit.'}  def recortar(numberToCut:'int',lowerLimit:'int',upperimit:'int')->rd:      try:          numberToCut = int(numberToCut)          lowerLimit = int(lowerLimit)          upperimit = int(upperimit)          if type(numberToCut)==int and type(lowerLimit)==int and type(upperimit)==int :              if (lowerLimit < numberToCut ) :                  return lowerLimit              elif (upperimit > numberToCut) :                  return upperimit              else :                  return numberToCut          else:              # assert False,"parameter numero #01 o numero #02 no son numero enteros"              return False      except ValueError:          print('Error, el número a recortar a recortar (parametro #01): {} , lowerLimit  (parametro #02): {} y upperimit  (parametro #03): : {} no son numeros enteros, debe cambiarlos para continuar '.format(numberToCut,lowerLimit,upperimit))          # assert False,"Error, el parametro de numero #1 : {} o el parametro numero #2 : {} no son numeros enteros, debe cambiarlos para continuar ".format(number01,number02)          return False    if \_\_name\_\_ == '\_\_main\_\_':      assert recortar(15,12,15)  == 12, 'Ok'      assert recortar(15,15,20)  == 20, 'Ok'      assert recortar(15,15,15)   == 15, 'Ok'        print("\*" \* 100)      print( recortar.\_\_annotations\_\_['return'] )      print ( recortar(15,0,10)   )      numberToCut = input(f"Ingrese valor a recortar :")      lowerLimit  = input(f"Ingrese valor del numero #1 para el Limite Inferior:")      upperimit   = input(f"Ingrese valor del numero #2 para el limite Superior:")      print(  recortar(numberToCut,lowerLimit,upperimit)  ) |

**6) Realiza una función separar() que tome una lista de números enteros y devuelva dos listas ordenadas. La primera con los números pares, y la segunda con los números impares:**

**🖐 Ayuda:** Para ordenar una lista automáticamente puedes usar el método .sort()

|  |  |
| --- | --- |
| Respuesta | rd={'listNumber':list,'list number':'integers',      'docstring':'odd and even list'}  def separar(listNumber:'list')->rd:      try:          if  listNumber and type(listNumber)==list :              if( all(map(  lambda p: str(p).isdigit(),listNumber))   ):                  list\_odd = [numero for numero in listNumber if numero %2 == 1]                  list\_even = [numero for numero in listNumber if numero %2 == 0]                  return [sorted(list\_even),sorted(list\_odd)]              else :                  print('Error, el parametro #1 listNumber: {} , contiene elmentos no numericos, debe modificarla para poder continuar '.format(listNumber))                  return False          else:              print('Error, el parametro #1 listNumber: {} , no es una lista '.format(listNumber))              return False      except ValueError:          print('Error, el parametro #1 listNumber: {} , contiene elmentos no numericos, debe modificarla para poder continuar '.format(listNumber))          return False  if \_\_name\_\_ == '\_\_main\_\_':      assert separar([1, 5, 7, 13, 22, 15, 26, 64, 34, 72, 52, 14])  == [[14, 22, 26, 34, 52, 64, 72],[1, 5, 7, 13, 15]], 'Ok'      assert separar(['a', 5, 7, 13, 22, 15, 26, 64, 34, 72, 52, 14])  == False, 'Ok'      print("\*" \* 100)      print( separar.\_\_annotations\_\_['return'] )      pares, impares = separar([1, 5, 7, 13, 22, 15, 26, 64, 34, 72, 52, 14])      print(pares)      print(impares) |