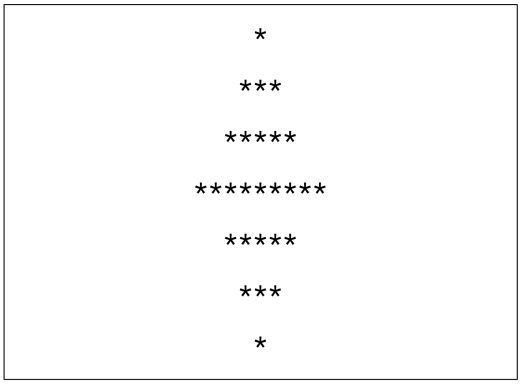
QUICES C&P

Valeria González González

QUIZ #1

Codifique 7 líneas de la función print() para producir un resultado como el de la imagen de abajo



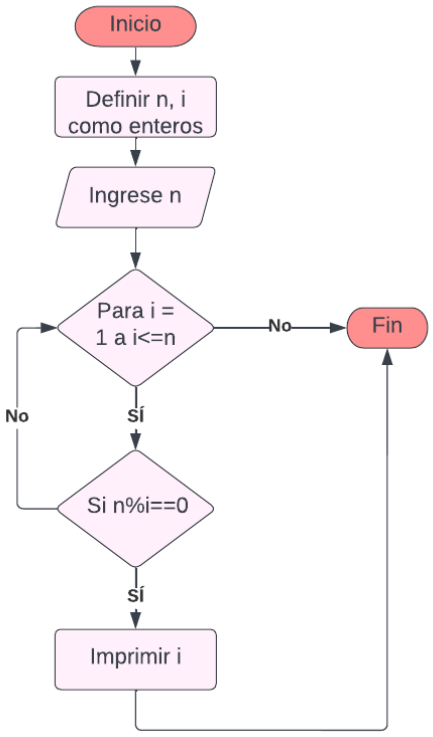

```
print(" "*3,"*"*1)
print(" "*2,"*"*3)
print(" ","*"*5)
print("*"*9)
print(" ","*"*5)
print(" "*2,"*"*3)
print(" "*3,"*"*1)
```

Output:

```
   *
  ***
 *****
*********
 *****
  ***
   *
```

QUIZ #2

Escriba un diagrama de flujos que factorice 18, 39, 63, 126, 792.



QUIZ #3

```
#Pedir datos a usuario
a = float(input("Ingrese número: "))
b = float(input("Ingrese otro número: "))

#Operaciones
suma = a + b
resta = a - b
prod = a * b
div = a / b
dive = a // b
mod = a % b
pot = a ** b

#Impresion
print("La suma de los datos ingresados es: ", suma)
print("La resta de los datos ingresados es: ", resta)
print("El producto de los datos ingresados es: ", prod)
print("La división de los datos ingresados es: ", div)
print("La división entera de los datos ingresados es: ", dive)
print("El módulo de los datos ingresados es: ", mod)
print("La potencia de los datos ingresados es: ", pot)
```

QUIZ #4

Use la función int para retornar el número 100 sumando el carácter "50" al número 50. Adicionalmente, use la función str para retornar el resultado de la adición "5050".

```
print(int("50") + 50)
print(str(50) + str(50))
```

```
100
5050
```

QUIZ #5

Se proveen una letra '1' y tres '0'. Use estas para construir el número 1000. Aquí solo se permiten operaciones de adición entre strings, y la función int() puede ser usada sólo una vez.

```
sumastr = "1" + "0" + "0" + "0"
print(int(sumastr))
```

```
1000
```

QUIZ #6

n! se define como n * (n-1) * (n-2) * (n-3) ... 2 * 1. Encuentre 5! y 10! usando enteros y el operador * e imprimirlos como sigue.

```
print("Calcule el factorial.")
f5 = 5 * (5-1) * (5-2) * (5-3) * (5-4)
print("5! = ", f5)
print("Calcule el factorial.")
f10 = 10 * (10-1) * (10-2) * (10-3) * (10-4) * (10-5) * (10-6) * (10-7)
* (10-8) * (10-9)
print("10! = ", f10)
```

```
Calcule el factorial.
5! =  120
Calcule el factorial.
10! =  3628800
```

QUIZ #7

IMC

```
# Pregunta el nombre del usuario
n = input("¿Cuál es tu nombre? ")
# Pregunta la estatura del usuario
e = float(input("¿Cuál es su estatura (en metros, con punto)? "))
# Pregunta el peso del usuario
p = float(input("¿Cuál es su peso (en kilogramos)? "))

# Calculo del indice de masa corporal del usuario
IMC = p / e**2

# Imprime el del indice de masa corporal del usuario
w = print(n,", su índice de masa corporal es de", IMC)

# Opcional
if 18.5 < IMC < 24.9 :
    print("Está en una condición saludable óptima")
else:
    print("No está en una condición saludable óptima")
```

QUIZ #8

Asigne los valores 30, 60 a las variables width y height, respectivamente. Escriba un programa que use esas dos variables para encontrar el área del rectángulo como se muestra abajo.

```
width = 30
height = 60

a = width * height

print("Area del Rectángulo: ", a)
```

QUIZ #9

El teorema de Pitágoras afirma que el cuadrado de la hipotenusa c para cualquier triángulo rectángulo es igual al cuadrado de la base a más el cuadrado de la altura b. Escriba un código que calcule el largo de la hipotenusa recibiendo la base y la altura como enteros.

```
a = int(input("Ingrese la base: "))
b = int(input("Ingrese la altura: "))
print("La longitud de la hipotenusa es: ", (a**2 + b**2)**(1/2))
```

QUIZ #10

Reciba el valor de un radio del usuario e imprima el área y la circunferencia de un círculo con dicho radio. Use la variable PI = 3.141592 para obtener esos valores.

```
PI = 3.141592
radio = float(input("Ingrese el valor del radio: "))
a = PI * radio * radio
c = 2 * radio * PI

print("La circunferencia del círculo es = ", c, ", el área del círculo
es = ", a)
```

```
Ingrese el valor del radio: 11
La circunferencia del círculo es =  69.115024 , el área del círculo es =  380.13263200000006
```

QUIZ #11

Escriba un programa que muestre los valores cuadrados de 2 a 6 en una tabla como la que se muestra a continuación. Como se muestra abajo, a se puede incrementar de 2 a 6, y n tiene el valor de 2. Ingrese el valor actual para la parte correspondiente a 'an', de tal forma que el resultado de salida de la ecuación sea 2**2.

```
n = 2
print("a", "\t", "n", "\t", "a ** n")
```

```
print(2, "\t", n, "\t", 2 ** n)
print(4, "\t", n, "\t", 4 ** n)
print(5, "\t", n, "\t", 5 ** n)
print(6, "\t", n, "\t", 6 ** n)
```

```
a       n       a ** n
2       2       4
4       2       16
5       2       25
6       2       36
```

QUIZ #12

Escriba un código que reciba el valor n a través de la entrada del teclado del usuario. Devuelve True si el entero n dado es impar y devuelve False si el entero es par. Para los casos en que n es 20 y 21, imprima lo siguiente...

```
n = float(input("Enter an integer: "))

if n % 1 == 0:
  if n % 2 == 0:
    print("Is the integer odd?:: ", False)
  else:
    print("Is the integer odd?:: ", True)
else:
  print("Enter an integer! >:c")
```

QUIZ #13

Escriba un código que tome la entrada del usuario y determine si el valor entero n es #un número par dentro del rango de 0 a 100 o no.

```
n = float(input("Enter an integer: "))

if n % 1 == 0:
  if (0 <= n <= 100):
    print("Is the input an even integer between 0 and 100?", False)
  else:
    print("Is the input an even integer between 0 and 100?", True)
else:
   print("Enter an integer! >:c")
```

QUIZ #14

Recibir un número entero de 3 dígitos del usuario. Si el centésimo digito del entero n es 3, devuelve True. Si no, devuelve Falso.

```
n = float(input("Enter a 3-digit integer: "))

if (n % 1 == 0):
    if n // 100 == 3:
      div = n // 100 == 3
      print(div)
    else:
      print(False)
else:
   print("Enter a 3-digit integer! >:c")
```

QUIZ #15

Recibe un entero. Si el número entero es múltiplo de 5, devuelve True. Si no, devuelve False

```
n = float(input("Enter an integer: "))

if (n % 1 == 0):
    if n % 5 == 0:
      cond = n % 5 == 0
      print(cond)
    else:
      print(False)
else:
   print("Enter a 3-digit integer! >:c")
```

QUIZ # 16

```
game_score = float(input("Enter game score: "))

if game_score > 1000:
  print("game_score = ", round(game_score))
  print("You are a master")
else:
  print("game_score = ", round(game_score))
```

QUIZ # 17

```
game_score = float(input("Enter game score: "))

print("game_score = ", round(game_score))
if game_score > 1000:
  print("You are a master")
else:
  pass
```

QUIZ # 18

```
x = float(input("Enter an integer between -100 and 100: "))

if (x % 1 == 0) and (-100 < x  and x < 100):
    if x > 0:
      print("X = ", round(x))
      print(x, "is a natural number")
    else:
      print("X = ", round(x))
else:
   print("Enter an integer between -100 and 100!")
```

QUIZ # 19

```
edad = float(input("Enter age: "))

if edad > 20:
  print("Adult")
if (edad <= 20) and (edad >= 10):
  print("Youth")
if edad < 10:
  print("Kid")
```

QUIZ # 20

```
edad = float(input("Enter age: "))
altura = float(input("Enter height in cm: "))

if (edad >= 18) and (altura > 150):
  print("You can enter")
else:
  print("You can't")
```

QUIZ # 21

```
print("Welcome to Yummy Restaurant. Here is the menu.")
print("- Burger (enter b)")
print("- Chicken (enter c)")
print("- Pizza (enter p)")

def menu():
  ch = input("Choose a menu (enter b, c, p): ")
  if ch == "b":
    print("You chose burger.")
  else:
    if ch == "c":
      print("You chose chicken.")
    else:
      if ch == "p":
        print("You chose pizza.")
      else:
        print("Try again")
        menu()
  return
menu()
```

UNIDAD 7

QUIZ # 1

Reciba una letra alfabética del usuario e imprima 'is a vocal' para a, e, i, o, u, y 'is a consonant' para cualquier otra letra

```
letra = input("Enter the alphabet: ")

if letra == 'a':
  print(letra, 'is a vocal')
elif letra == 'e':
  print(letra, 'is a vocal')
elif letra == 'i':
  print(letra, 'is a vocal')
elif letra == 'o':
  print(letra, 'is a vocal')
elif letra == 'u':
  print(letra, 'is a vocal')
else:
  print(letra, 'is a consonant')
```

QUIZ #2

Escriba el siguiente programa que recibe dos número enteros a, b como entrada, determina si a es un múltiplo de b e imprime el resultado

```
a, b = input("Ingrese dos números enteros (separados por espacio):
").split()

a = int(a)
b = int(b)

if a % b == 0:
  print(a, 'es múltiplo de ', b)
else:
  print(a, 'no es múltiplo de ', b)
```

QUIZ # 1

Escriba un programa que ejecute sumas, restas, multiplicaciones y divisiones. Imprime el resultado de la operación de dos números enteros positivos, en función del número de operación deseado dado como entrada. Si se ingresa un número diferente a 1,2,3 y 4, se imprime "Se ingresó un número incorrecto". Para ingresar dos números, escribe uno, presiona enter y escribe otro

```
print("1) Addition","\t", "2) Subtraction","\t", "3)
Multiplication","\t", "4) Division")

operation = float(input("Enter the desired number of operation: "))

if (operation == 1) or (operation == 2) or (operation == 3) or
(operation == 4):
  print("Enter two numbers for operation. ")
  a = float(input(""))
  b = float(input(""))
  if operation == 1:
    result = a + b
    print(round(a), " + ", round(b), " = ", round(result))
  elif operation == 2:
    result = a - b
    print(round(a), " - ", round(b), " = ", round(result))
  elif operation == 3:
    result = a * b
    print(round(a), " * ", round(b), " = ", round(result))
  elif operation == 4:
    result = a / b
```

```
    print(round(a), " / ", round(b), " = ", round(result))
else:
  print("Entered an incorrect number.")
```

QUIZ #4

Escriba un programa que reciba un punto con coordenadas x e y como entrada y determine a qué cuadrante entre 1, 2, 3, 3 pertenece el punto. La posición del cuadrante se muestra enla siguiente figura

```
x, y = input("Enter x, y coordinates (comma separated): ").split()
x = float(x)
y = float(y)

if (x > 0) and (y > 0):
  print("In the first quadrant")
elif (x < 0) and (y > 0):
  print("In the second quadrant")
elif (x < 0) and (y < 0):
  print("In the third quadrant")
elif (x > 0) and (y < 0):
  print("In the fourth quadrant")
else:
  print("The coordinate is on an axis")

#elif (x = 0) or (y = 0):
#  print("The coordinate is on an axis")
```

QUIZ #3

Desarrolle un programa de pedidos del menú para Yummy Restaurant. Muestre el siguiente menú al usuario y permita que el usuario seleccione uno. Si el alfabeto de entrada dado no está en el menú, imprima 'ingrese al menú de nuevo:' y reciba otra entrada.

```
print("Welcome to Yummy Restaurant. Here is the menu.")
print("- Burger (enter b)")
print("- Chicken (enter c)")
print("- Pizza (enter p)")

def menu():
  ch = input("Choose a menu (enter b, c, p): ")
  if ch == "b":
```

```
    print("You chose burger.")
  elif ch == "c":
      print("You chose chicken.")
  elif ch == "p":
      print("You chose pizza.")
  else:
      print("Try again")
      menu()
  return
menu()
```

UNIDAD 17 - Q1. *(tiempo estimado: 5 min)*

Defina una función llamada *my_greet* que imprima "Bienvenido." y llame esta función para que imprima el saludo dos veces.

```
def my_greet(n = 2):
  print('Bienvenido\n' * n)
my_greet()
```

UNIDAD 17 - Q2. *(tiempo estimado: 10 min)*

Implemente la función *max2(m,n)* la cual tome dos parámetros llamados m y n y que regrese el número mayor de estos dos valores. Por otro lado, implemente la función *min2(m,n)* que tome dos parámetros llamados m y n y que regrese el número menor de estos dos valores. Luego, asigne el valor de 100 y 200 como argumentos para las dos funciones e imprima su resultado.

```
def max2(m,n):
  if m > n:
    return m
  return n


def min2(m,n):
  if m < n:
    return m
  return n


print('Valor minimo: ', min2(100, 200), '\nValor máximo: ', max2(100,
200))
```

UNIDAD 17 - Q3. *(tiempo estimado: 5 min)*

Queremos cambiar el valor de la distancia de millas a kilómetros (Las millas son una unidad de medida que se usa principalmente en Estados Unidos, mientras que los kilómetros son la unidad estándar internacional). Para ello implemente la función *mile2km(ml)* que toma valores de las millas y las convierta en kilómetros. Luego llame a esta función para que imprima la conversión de las millas 1 a 5 en kilómetros a través de la sentencia *for*. (Se define 1 milla con 1.61 km)

```
def mile2km(ml):
  km = ml * 1.61
  return km


for i in range(1,6):
  print('{} ml equivale a {} km'. format(i, mile2km(i)))
```

UNIDAD 17 - Q4. *(tiempo estimado: 5 min)*

Implemente la función *cel2fah(cel)* que toma la temperatura en grados Celsius y que retorna la temperatura en grados Fahrenheit. Luego llame la función, a través de una sentencia *for*, que imprima las conversiones de temperatura desde 10 hasta 50 grados Celsius en pasos de 10 unidades. (Recuerde la fórmula de conversión Fah = Cel*(9/5) + 32

```
def cel2fah(cel):
  fah = cel * (9 / 5) + 32
  return fah

for i in range(10, 51, 10):
  print('{} °C equivale a {} F'. format(i, cel2fah(i)))
```

UNIDAD 17 - QUIZ 1

Permita que un usuario ingrese 3 enteros a, b y c e imprima el valor promedio, el máximo y el mínimo de estos 3. Para ello, defina y llame las funciones *mean3(a, b, c)*, *max3(a, b, c)*, *min3(a, b, c)*.

Debes tener una salida similar a la que se muestra a continuación:

```
def mean3(a, b, c):
  return (a + b + c) / 3

def max3(a, b, c):
  return max(a, b, c)
```

```
def min3(a, b, c):
  return min(a, b, c)
a, b, c = [int(i) for i in input('Enter three numbers: ').split()]

print('\nThe average value of {}, {}, {} is {}'. format(a, b, c,
mean3(a, b, c)))
print('The maximum value of {}, {}, {} is {}'. format(a, b, c, max3(a,
b, c)))
print('The minimum value of {}, {}, {} is {}'. format(a, b, c, min3(a,
b, c)))
```

UNIDAD 18 - Q1. *(tiempo estimado: 5 min)*

Toma un número n como entrada y encuentra su suma desde 1 hasta n. Escriba su función usando funciones recursivas.

```
def suma(n):
  if n == 0:
    return 0
  else:
    return n + suma(n-1)


suma(1)
```

UNIDAD 18 - Q2. *(tiempo estimado: 5 min)*

Python tiene el operador ** el cual indica una potencia, Sin embargo, tomemos a x y n como entradas sin usar ese operador y use una función recursiva que retorne x a la potencia de n. Luego, calcule 2 10 poniendo x como 2 y n como 10.

```
def potencia(x, n):
  if n == 1: #0
    return x #1
  else:
    return x * potencia(x, n-1)

potencia(2, 10)
```

UNIDAD 18 - Q1.

El número natural e, también llamado el número de Euler o la constante de Napier, en un número irracional usado como la base de los logaritmos naturales y es definido con la siguiente fórmula:

```
def factorial(n):
    if n <= 1:
        return 1
    else:
        return n * factorial(n-1)
def euler(n):
  if n == 0:
    return 1
  else:
    return (1 / factorial(n)) + euler(n - 1)

print('{:.5f}'.format(euler(20)))
```

UNIDAD 19 - Q1.

Hay una lista de elementos enteros llamada n_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Retorne una lista llamada *even_list* la cual solo contiene los elementos pares de n_list usando las funciones *filter* y *lambda*.

```
n_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_list = []
for a in filter(lambda x: x % 2 == 0, n_list):
  even_list.append(a)
print('even_list =', even_list)
```

UNIDAD 19 - Q2.

Hay una lista de elementos enteros llamada n_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Retorne una lista llamada *even_list* la cual solo contiene los elementos pares de n_list usando las funciones *filter* y *lambda*. Sin embargo, en este ejercicio no use el ciclo for y en vez de ello, use una función *list*.

```
n_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_list = list(filter(lambda x: x % 2 == 0, n_list))
print(even_list)
```

UNIDAD 19 - Q3.

Escriba una función *map* que convierta la variable 'a_list', que contiene el alfabeto en minúscula ['a', 'b', 'c', 'd'] y lo convierte en la variable 'upper_a_list', que contiene el alfabeto en mayúscula ['A', 'B', 'C', 'D']. También defina una función llamada to_upper que reciba las letras en minúsculas como parámetros y las retorne como letras en mayúsculas.

```
def to_upper(x):
  return x.upper()

a_list = ['a', 'b', 'c', 'd']
upper_a_list = list(map(to_upper, a_list))
print('upper_a_list =', upper_a_list)
```

UNIDAD 19 - Q4.

Calcule la suma de enteros de 1 a 100 usando la función *reduce* y la expresión *lambda* dentro de ella. Use como entrada: range(1, 101).

```
from functools import reduce

suma = reduce(lambda x, y: x + y, range(1, 101))
print('La suma de 1 a 100 es:', suma)
```

UNIDAD 19 - Q1.

Tienes disponible la información de los puntajes de los exámenes de los estudiantes en inglés, matemáticas y ciencias que pueden ser expresados como una lista [100, 90, 95]. Si hay dos estudiantes, los puntajes se representarían como [100, 90, 95, 90, 85, 93]. Ahora, si un estudiante no realizó algún examen en particular se marca con el puntaje de 0.

```
scores = [100, 90, 95, 90, 80, 70, 0, 80, 90, 90, 0, 90, 100, 75, 20,
30, 50, 90]

separados = [scores[i:i + 3] for i in range(0, len(scores), 3)]
#validos = list(filter(lambda estudiante: all(nota != 0 for nota in
estudiante), separados))
validos = list(filter(lambda estudiante: all(estudiante), separados))
print('scores = ', scores)
print('El número total de estudiantes es ', round(len(scores)/3))
print('El número total de estudiantes válidos es ', len(validos))
print(validos)
```

UNIDAD 20 - Q1.

Cree una función anidada llamando a esta función *greetings* y otra función llamada *say_hi* dentro de esa función. Llame la función *say_hi* dentro de la función *greetings* y luego, llame la función *greetings* e imprima 'hello'.

```
def greetings():
  def say_hi():
    print('hello')
  say_hi()
greetings()
```

UNIDAD 20 - Q2.

Escriba la siguiente función *calc* y asigne calc a la variable num. Luego, ejecute num(3), el resultado debe ser 14.

```
num = calc()
num(3)
```

UNIDAD 20 - Q3.

Reconstruya la funcion interna *mul_add* de la función anidada calc del problema anterior usando la expresion lambda y obtenga el mismo resultado.

```
def calc():
  a = 3
  b = 5
  return lambda x: a*x + b
```

UNIDAD 20 - Q1.

Extraiga la lista resultante de la lista *lst* que tiene valores de 1 a 100. La lista resultante tiene los elementos de *lst* que son divisibles por 5 o 7. Para ello declare la función func1(a) y anide las funciones func2 y func3. Luego llame las dos funciones desde la función func1 e imprima los números que son divisibles por 5 o 7. Finalmente, alinee los valores resultantes usando la función pre-construida *sorted()*.

```
lst = list(range(1,101))
print('lst = ', lst)
def func1(a):
  def func2():
      result1 = []
      for i in a:
          if i%5 == 0:
              result1.append(i)
      return result1
  def func3():
```

```
        result2 = []
        for i in a:
            if i%7 == 0:
                result2.append(i)
        return result2
    return sorted(func2()+func3())
print('result = ', func1(lst))
```

UNIDAD 21 - Q1.

Construya la clase *Dog* y sus objetos con sus funcionalidades descritas a continuación:

```
class Dog:
    def bark(self):
      print('woof woof')
my_dog = Dog()
my_dog.bark()
```

UNIDAD 21 - Q2.

Defina la clase Dog con las funcionalidades descritas a continuación y llam sus métodos e instancias:

```
class Dog:
  def __init__(self, name):
    self.__name = name
  def bark(self):
    print(self.__name,': woof woof')
my_dog = Dog('Bingo')
my_dog.bark()
```

UNIDAD 21 - Q1.

Construya la clase Studen que tiene las siguientes funcionalidades:

- Este estudiante toma quices para inglés, matemáticas, ciencias y los puntajes son dados como entradas. Genere instancias a través del uso de esta clase. Esta clase tiene los siguientes atributos y acciones

```
class Student:
  def __init__(self, name, id):
    self.__name = name
    self.__id = id
```

```
    self.__eng_quiz = 0
    self.__math_quiz = 0
    self.__science_quiz = 0

  def set_eng_quiz(self, score):
    self.__eng_quiz = score
  def set_math_quiz(self, score):
    self.__math_quiz = score
  def set_science_quiz(self, score):
    self.__science_quiz = score

  def get_name(self):
    return self.__name
  def get_student_id(self):
    return self.__id
  def get_eng_quiz(self):
    return self.__eng_quiz
  def get_math_quiz(self):
    return self.__math_quiz
  def get_science_quiz(self):
    return self.__science_quiz
  def get_total_score(self):
    return (self.get_eng_quiz() + self.get_math_quiz() +
self.get_science_quiz())
  def get_avg_score(self):
    return self.get_total_score()/3

  def __str__(self):
    return ("Name : {}, ID : {}".format(self.get_name(),
self.get_student_id()))
```

```
name = input("Enter the student's name : ")
def validacion_id(i):
    try:
        if len(str(int(i))) == 8:
          return i
        else: print("Enter a valid ID (8 digits)!")
    except:
        print("Enter a valid ID (Integer)!")

while True:
    id = validacion_id(input("Enter the student's ID : "))
    if id != None:
      break

student = Student(name, id)

# subj = ["English", "Mathematics", "Science"]
# for i in range(len(subj)):
#   score = float(input("Enter the student's", subj[i], "quiz score :
"))
#   subj[i] = score
english = float(input("Enter the student's English quiz score : "))
maths = float(input("Enter the student's Mathematics quiz score : "))
science = float(input("Enter the student's Science quiz score : "))

student.set_eng_quiz(english)
student.set_math_quiz(maths)
student.set_science_quiz(science)

print(student)
print(' English quiz score: ', student.get_eng_quiz())
print(' Mathematics quiz score: ', student.get_math_quiz())
```

```
print(' Science quiz score: ', student.get_science_quiz())
print('Total: ', student.get_total_score(), end = '  ')
print('Avarage: ', student.get_avg_score())
```

UNIDAD 22 - QUIZ 1:

Escriba la salida de los códigos del siguiente ejemplo que usa una pila.

```
stack=Stack()
stack.push('Banana') #['Banana']
stack.push('Apple') #['Banana', 'Apple']
stack.push('Tomato') # Tomato #['Banana', 'Apple', 'Tomato']
stack.pop() #['Banana', 'Apple']
stack.push('Strawberry') #['Banana', 'Apple', 'Strawberry']
stack.push('Grapes') #['Banana', 'Apple', 'Strawberry', 'Grapes']
stack.pop() # Grapes #['Banana', 'Apple', 'Strawberry']
print(stack.stack) #['Banana', 'Apple', 'Strawberry']
```

UNIDAD 22 - QUIZ 2:

Escriba la salida de los códigos del siguiente ejemplo que usa una pila.

```
stack=Stack()
items=[10*i for i in range(1,10)]
for item in items:
  stack.push(items)
  if (item // 10) % 2 == 0:
    stack.pop()
print(stack.stack) #[[10, 20, 30, 40, 50, 60, 70, 80, 90], [10, 20, 30,
40, 50, 60, 70, 80, 90], [10, 20, 30, 40, 50, 60, 70, 80, 90], [10, 20,
30, 40, 50, 60, 70, 80, 90], [10, 20, 30, 40, 50, 60, 70, 80, 90]]
```

UNIDAD 22 - QUIZ 1:

El documento HTML consta de muchas etiquetas, como se muestra a continuación. Escriba un programa que coincida con las etiquetas de documentos HTML.

```
text = input('Ingrese la cadena del documento HTML: ')
start = text.find('<')
while start != -1:
  end = text.find('>', start + 1)
  tag = text[start:end + 1]
  print(tag, end = ' ')
  start = text.find('<', end + 1)
```

UNIDAD 23 - QUIZ 1:

Escriba la salida de los códigos del ejemplo que usa una cola.

```
queue=Queue()
queue.enqueue('Banana') #['Banana']
queue.enqueue('Apple') #['Banana', 'Apple']
queue.enqueue('Tomato') #['Banana', 'Apple', 'Tomato']
queue.dequeue() #['Apple', 'Tomato']
queue.enqueue('Strawberry') #['Apple', 'Tomato', 'Strawberry']
queue.enqueue('Grapes') #['Apple', 'Tomato', 'Strawberry', 'Grapes']
queue.dequeue() #['Tomato', 'Strawberry', 'Grapes']
print(queue.queue)
```

UNIDAD 23 - QUIZ 2:

Escriba la salida de los códigos del ejemplo que usan una cola.

```
queue=Queue()
items=[10*i for i in range(1,11)]
for item in items:
  queue.enqueue(items)
  if (item // 10) % 2 == 0:
    queue.dequeue()
print(queue.queue) # Lista con 5 sublistas, c/u corresponde a items.
Elimina las sublistas correspondientes a elementos pares en items
# [[10, 20, 30, 40, 50, 60, 70, 80, 90, 100], [10, 20, 30, 40, 50, 60,
70, 80, 90, 100], [10, 20, 30, 40, 50, 60, 70, 80, 90, 100], [10, 20,
30, 40, 50, 60, 70, 80, 90, 100], [10, 20, 30, 40, 50, 60, 70, 80, 90,
100]]
```

UNIDAD 23 - QUIZ 1:

Consulte la definición de clase de la siguiente manera para completar y probar la clase Deque.

```
class Deque:
  def __init__(self):
    self.queue =[]

  def is_empty(self):
    return True if len(self.queue)==0 else False

  def add_first(self,item):
    self.queue.insert(0, item)
```

```
  def remove_first(self):
    return None if self.is_empty() else self.queue.pop(0)

  def add_last(self,item):
    self.queue.append(item)

  def remove_last(self):
    return None if self.is_empty() else self.queue.pop()

#Instanciando la clase en un objeto
cola_doble = Deque()
#Probando el método add_first
cola_doble.add_first("S")
cola_doble.add_first("N")
print(cola_doble.queue)
#Probando el método remove_first
cola_doble.remove_first()
print(cola_doble.queue)
#Probando el método add_last
cola_doble.add_last("I")
cola_doble.add_last("O")
print(cola_doble.queue)
#Probando el método remove_last
cola_doble.remove_last()
print(cola_doble.queue)
```

UNIDAD 24 - QUIZ 1:

Adivina la salida del siguiente algoritmo. Analice el código y describa el comportamiento de este algoritmo.

```
def find_two(nums):
  x=y=0
  for i in range(1, len(nums)):
    if nums[x]<nums[i]:
      x=i
    elif nums[y]>nums[i]:
      y=i
  return x,y #X indice del numero mayor, Y indice del numero menor

nums=[11,37,45,26,59,28,17,53]
i,j=find_two(nums)
print(nums[i],nums[j]) #nums[4],nums[0] # 59 11
```

UNIDAD 24 - QUIZ 2:

¿Cuántas comparaciones se deben hacer para find_two() que se implementó en el quiz #1?

Se recorre la lista inicial (len(nums)-1 veces) para tomar dichos elementos y compararlos en c/parte del condicional con su respectivo

UNIDAD 24 - QUIZ 1:

Escriba un programa para contar cuántas veces se usó una palabra específica en la oración ingresada por el usuario.

En la oración ingresada por el usuario, una palabra se distingue por la presencia de espacios.

- Use la función input().split() para crear la lista S que tiene cada cadena como su elemento.
- Use la función input() para recibir la palabra que el usuario buscará y guárdela en x.
- La función word_count() recibe S y x como parámetros y proporciona un retorno contando cuántas x están presentes en la S.

```
def word_count(s, x):
  count = 0
  for i in s:
    if i == x:
      count += 1
    else:
      0
  return count


S = list(input('Ingrese una oración: ').split())
x = input('Ingrese la palabra a buscar: ')
count = word_count(S,x)
print(f'En la lista S, la palabra {x} aparece {count} veces.')
```

UNIDAD 25 - QUIZ 1:

El siguiente es el código para el juego de "adivinar números". Si maximum = 100 y number = 51, ¿cuántos conteos se imprimirían?

```
from random import randint
```

```
maximum=int(input('Ingrese el número máximo: '))
number=int(input('Ingrese su número a adivinar: '))
count=0
low,high=1,maximum
while low<high:
  mid=(low+high)//2
  count +=1
  if mid==number:
    print(f'Su número es {number}.')
    break
  elif mid>number:
    high=mid-1
  else:
    low=mid+1
print(f'Se tiene un total de {count} veces.') #6
```

UNIDAD 25 - QUIZ 2:

Si el máximo = 100 y el número = 25 en el juego de adivinanzas, ¿cuántos conteos se imprimirán?

```
#2
```

UNIDAD 25 - QUIZ 1:

Si se proporciona una lista ordenada de números y un número entero aleatorio x, implemente la función que devuelve la ubicación del índice donde se insertará x. Sin embargo, S debe ser una lista ordenada incluso después de insertar x.

```
def search_insert_position(nums,x):
  low = 0
  high = len(nums) - 1
  while low <= high:
    mid = (low + high) // 2
    if nums[mid] == x:
      return mid
    elif nums[mid] < x:
```

```
      low = mid + 1
    else:
      high = mid - 1
  return low

nums=[10,20,40,50,60,80]
x=int(input('Ingrese un número para ser insertado: '))
pos=search_insert_position(nums,x)
print(f'x={x} debe ser insertado en la posición {pos}.')
nums.insert(pos,x)
print(nums)
```

UNIDAD 26 - QUIZ 1:

Utilice la función hash de la tabla hash para calcular la clave y la clave hash de "Alice in Wonderland".

```
table=HashTable(8)
book='Alice in Wonderland'
key=sum(map(ord,book))
clave_hash = table.hash(key)
#print(key, clave_hash, table.table[clave_hash])
print(key, clave_hash, table.get(key))
```

UNIDAD 26 - QUIZ 1:

Si hay 10 ranuras en la estantería, con el siguiente código encuentre las ranuras donde se colocarán cada uno de los siguientes libros.

```
table=HashTable(10)
books=[
    'The Little Prince',
    'The Old Man and the Sea',
    'The Little Mermaid',
    'Beauty and the Beast',
    'The Last Leaf',
    'Alice in WonderLand'
]
for book in books:
  key=sum(map(ord,book))
  table.put(key,book)
```

```
for key in table.table.keys():
  print(key,table.table[key]) #Ranuras 3, 4, 7, 8 y 9
```

UNIDAD 26 - QUIZ 1:

Cree un convertidor que convierta números arábigos en números romanos utilizando la tabla hash que se proporciona a continuación.

```
table={1000:'M', 900:'CM', 500:'D', 400:'CD',
       100:'C', 90:'XC', 50:'L', 40:'XL',
       10:'X', 9:'IX', 5:'V', 4:'IV', 1:'I'}
def int_to_roman(num:int):
  result = ""

  # comparando miles
  miles = num // 1000
  if miles != 0:
    result = miles * table[1000]
    num -= miles * 1000

  # comparando centenas
  cents = num // 100
  if cents != 0:
    if 1 <= cents < 4:
      result += cents * table[100]
    elif cents == 4:
      result += table[400]
    elif cents == 5:
      result += table[500]
    elif 5 < cents <= 8:
      result += table[500] + cents * table[100]
    else:
      result += table[900]
    num -= cents * 100

  # comparando decenas
  dec = num // 10
  if dec != 0:
    if 1 <= dec < 4:
      result += dec*table[10]
    elif dec == 4:
      result += table[40]
    elif dec == 5:
```

```
      result += table[50]
    elif 5 < dec <= 8:
      result += table[50] + dec*table[10]
    else:
      result += table[90]

  # comparando unidades
  uni = num % 10
  if uni != 0:
    if 1 <= uni < 4:
      result += uni * table[1]
    elif uni == 4:
      result += table[4]
    elif uni == 5:
      result += table[5]
    elif 5 < uni <= 8:
      result += table[5] + uni * table[1]
    else:
      result += table[9]
  return result

num=int(input('Ingrese un número: '))
print(int_to_roman(num))

# def int_to_roman(num:int) -> str:
#   roman = ''
#   symbols = ['M', 'CM', 'D', 'CD', 'C', 'XC', 'L', 'XL', 'X', 'IX',
'V', 'IV', 'I']
#   values = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]
#   for i, v in enumerate(values):
#     while num >= v:
#       roman += symbols[i]
#       num -= v
#   return roman

# num = int(input('Ingrese un número: '))
# print(int_to_roman(num))



-------------------------------------------------------------
def int_to_roman(num:int):
#   result = ""
```

```
#   # comparando miles
#   miles = num // 1000
#   if miles != 0:
#     result = miles*table[1000]
#     num -= miles*1000

#   #Creando lista para el ciclo for
#   list_unit = [100, 10, 1]
#   for unit in list_unit:
#     if unit != 1:
#       selected_unit = num // unit
#     else:
#       selected_unit = num

#   # comparando la unidad dependiendo de la lista
__import__("ipdb").set_trace()
#     if selected_unit != 0:
#       if 1 <= selected_unit < 4:
#         result += selected_unit*table[unit]
#       elif selected_unit == 4:
#         result += table[4*unit]
#       elif selected_unit == 5:
#         result += table[5*unit]
#       elif 5 < selected_unit <= 8:
#         result += table[5*unit] + selected_unit*table[unit]
#       else:
#         result += table[9*unit]
#       num -= selected_unit*unit
#   return result
```

def busqueda_binaria_nombres(nombres, nombre_buscado): izquierda = 0 derecha = len(nombres) - 1 while izquierda <= derecha: medio = (izquierda + derecha) // 2 if nombres[medio] == nombre_buscado: return medio elif nombres[medio] < nombre_buscado: izquierda = medio + 1 else: derecha = medio - 1 return -1 nombres = ["Ana", "Carlos", "Juan", "Luisa", "Pedro", "Victor"] indice_juan = busqueda_binaria_nombres(nombres, "Juan") if indice_juan == -1: print("El nombre no fue encontrado") else: print("El nombre fue encontrado en la posición", indice_juan)

UNIDAD 27 - QUIZ 1:

¿Cuántas operaciones de intercambio se han ejecutado en el proceso de ordenamiento de burbujas siguiente?

```
def burbuja(S):
    n = len(S)
    count = 0
    for i in range(n):
        print(S)
        for j in range(n -1):
            if S[j] > S[j +1]:
                S[j], S[j + 1] = S[j + 1], S[j]
                count += 1 #🔵
    print('Se hicieron {} intercambios', format(count)) #8

S = [50, 30, 40, 10, 20]
burbuja(S)
print(S)
```

UNIDAD 27 - QUIZ 2:

¿Cuántas operaciones de comparación se han ejecutado en el proceso de ordenamiento por inserción que se muestra a continuación?

```
def clasificación_por_inserción2(S):
    n = len(S)
    count = 0
    for i in range(1, n):
        print(S)
        x = S[i]
        j = i - 1
        while j >= 0 and S[j] > x: #🔵
            count += 1
            S[j + 1] = S[j]
            j -= 1
        S[j + 1] = x
    print('Se hicieron {} comparaciones', format(count)) #8
S = [50, 30, 40, 10, 20]
clasificación_por_inserción2(S)
print(S)
```

UNIDAD 27 - QUIZ 1:

Dadas dos palabras, escribe un algoritmo que determine si estas dos cadenas son anagramas. Un anagrama es una palabra que se forma reordenando las letras de otra palabra utilizando todas las letras originales exactamente una vez. (Por ejemplo, "ESCUCHAR" y "SILENCIO" son anagramas).

```
def insertion_sort2(S):
    n = len(S)
    for i in range(1, n):
        x = S[i]
        j = i -1
        while j >= 0 and S[j] > x:
            S[j + 1] = S[j]
            j -= 1
        S[j + 1] = x
def son_anagramas():
  palabra1 = str(input("Ingrese palabra 1: "))
  palabra2 = str(input("Ingrese palabra 2: "))

  count = 0
  while count == 0:
    if not palabra1.isalpha():
      palabra1 = str(input("Ingrese únicamente letras (a-z) en la
palabra 1: "))
    else:
      count = 1

  count = 0
  while count == 0:
    if not palabra2.isalpha():
      palabra2 = str(input("Ingrese únicamente letras (a-z) en la
palabra 2: "))
    else:
      count = 1

  if len(palabra1) == len(palabra2):
    palabra1 = palabra1.lower()
    palabra2 = palabra2.lower()

    palabra1_ordenada = insertion_sort2(list(palabra1))
    palabra2_ordenada = insertion_sort2(list(palabra2))

    if palabra1_ordenada == palabra2_ordenada:
```

```
        return True
    return False

son_anagramas()
```

UNIDAD 27 - QUIZ 2:

El uso de un algoritmo de clasificación permite determinar fácilmente si se trata de un anagrama o no.

- Cree una función que evalúe los anagramas utilizando la función sorted() incorporada de Python.
- Modificar la función selection_sort2() para crear una función que determine los anagramas.
- Modificar la función insertion_sort2() para crear una función que determine el anagrama.

```
word1 = input("Ingrese la primer palabra: ")
word2 = input("Ingrese la segunda palabra: ")

def is_anagram_sorted(x, y):
    if len(x) == len(y)
        return sorted(x.upper()) == sorted(y.upper())
    return False
print(is_anagram_sorted(word1, word2))

def selección_de_orden2(S):
    n = len(S)
    for i in range (n - 1):
        mas_pequeño = i
        for j in range(i + 1, n):
            if S[j] < S[mas_pequeño]:
                mas_pequeño = j
        S[i], S[mas_pequeño] = S[mas_pequeño], S[i]

def is_anagram_selection(x, y):
    return selección_de_orden2(list(x)) == selección_de_orden2(list(y))
print(is_anagram_selection(word1, word2))
def clasificación_por_inserción2(S):
    n = len(S)
    for i in range(1, n):
        x = S[i]
        j = i -1
```

```
        while j >= 0 and S[j] > x:
            S[j + 1] = S[j]
            j -= 1
        S[j + 1] = x
def is_anagram_insertion(x, y):
    return clasificación_por_inserción2(list(x)) ==
clasificación_por_inserción2(list(y))
print(is_anagram_insertion(word1, word2))
```

UNIDAD 28 - QUIZ 1:

¿Cuántas veces se ejecutó la función ordenamiento_fusion2() en el proceso de ordenamiento por mezcla que se muestra a continuación?

```
count = 0
def ordenamiento_fusion2(S, low, high):
    global count
    count += 1
    if low < high:
        print(S)
        mid = (low + high) // 2
        ordenamiento_fusion2(S, low, mid)
        ordenamiento_fusion2(S, mid + 1, high)
        fusion2(S, low, mid, high)

S = [6, 2, 11, 7, 5, 4, 8, 16, 10, 3]
ordenamiento_fusion2(S, 0, len(S)-1)
print(S)
print(count) #19

count = 0
S = [6, 2, 11, 7, 5, 4, 8, 16, 10, 3, 1, 12, 9]
ordenamiento_fusion2(S, 0, len(S)-1)
print(S)
print(count) #25
```

UNIDAD 28 - QUIZ 1:

Dadas N listas ordenadas como entrada, escribe un programa que las fusione en una lista ordenada.

```
def fusion1(L, R):
  S = []
```

```
  while len(L)>0 and len(R)>0:
    if L[0] <= R[0]:
      S.append(L.pop(0))
    else:
      S.append(R.pop(0))
  while len(L) != 0:
    S.append(L.pop(0))
  while len(R) != 0:
    S.append(R.pop(0))
  return S

def fusion_multiple(lista_de_numeros: list): #lista final
  lista_ord = []
  for i, lista in enumerate(lista_de_numeros):
    print(lista_ord)
    if i == 0:
      lista_ord = lista
    else:
      lista_ord = fusion1(lista_ord, lista)
  return lista_ord

N = int(input("Ingrese el número de lista: "))
lista_de_numeros = []
for i in range(N):
    numeros = list(map(int, input("Ingrese una lista de números:
").split()))
    print(numeros)
    lista_de_numeros.append(numeros)
ordenada = fusion_multiple(lista_de_numeros)
print("Fusionada dentro : ", ordenada)
```

UNIDAD 29 - QUIZ 1:

Dada la siguiente lista, escriba la salida después de ejecutar la función particion1()

```
S = [15, 10, 12, 20, 25, 13, 22]
# [13, 10, 12, 15, 25, 20, 22]
ordenamiento_rapido1(S, 0, len(S) - 1)
print(S)
```

UNIDAD 29 - QUIZ 1:

Escriba un algoritmo que encuentre el Kº elemento mayor, dados N elementos desordenados.

Después de resolver el problema con los dos métodos anteriores, analiza qué algoritmo es más eficiente.

- Puede devolver el elemento Kth después de utilizar la función de ordenamiento.
- Puede utilizar la función partition() para realizar una llamada recursiva hasta que el pivote sea el Kº elemento.

```
def k_esimo1(S, k):
  ordenamiento_rapido2(S, 0, len(S) -1)
  print(S)
  return S[-k]
S = [15, 10, 12, 20, 25, 13, 22]
k_esimo1(S, 2)

k_number = 0

----------
def particion2(S, low, high):
    rand = randint(low, high)
    S[low], S[rand] = S[rand], S[low]
    pivote, izquierda, derecha = S[low], low, high
    print(S, izquierda, derecha, "Pivote = ", pivote)
    while izquierda <derecha:
        while izquierda < high and S[izquierda] <= pivote:
            izquierda += 1
        while derecha > low and pivote <= S[derecha]:
            derecha -= 1
        if izquierda < derecha:
            S[izquierda], S[derecha] = S[derecha], S[izquierda]
    S[low], S[derecha] = S[derecha], S[low]
    return derecha

def k_esimo2(S, k):
  n = len(S)
  if k <= 0 or k > n:
    return None # K es inválido
  low, high = 0, n - 1
  while low < high:
    pivot = particion2(S, low, high)
    if pivot == n - k:
```

```
      return S[pivot]
    elif pivot > n - k:
      high = pivot - 1
    else:
      low = pivot + 1
  return None # K es inválido

S = [15, 10, 12, 20, 25, 13, 22]
k_esimo2(S, 2)
```

UNIDAD 30 - QUIZ 1:

En el problema del intercambio de monedas, supongamos que hay una moneda de 400 wones.

Si es así, escriba el resultado de cómo el algoritmo coin_change() determinará el cambio de la moneda de 800 wones.

```
coins = [500, 400, 100, 50, 10]
amount = int(input('Ingrese la cantidad:'))
changes = coin_change(coins, amount)
print('Ingrese la cantidad:', amount)
print(changes, len(changes)) # [500, 100, 100, 100] 4
```

UNIDAD 30 - QUIZ 1:

Suponga que el número de monedas en el problema de intercambio de monedas no es infinito. Por ejemplo, si tiene las siguientes monedas en su billetera, ¿cómo debe distribuir 710 won como cambio? #`[500, 100, 50, 50, 10]`

```
def coin_change2(coins:list, amount:int):
  if amount <= 0:
    return "No se necesita entregar cambio"
  changes = []
  largest = 0
  try:
    while(amount > 0):
      if (amount >= coins[largest][0]) & (coins[largest][1] != 0):
        changes.append(coins[largest][0])
        amount -= coins[largest][0]
        coins[largest][1] -= 1
      else:
        largest += 1
    return changes
  except:
```

```
    return "No hay cambio disponible"

coins = [[500, 1] , [100, 1] , [50, 3 ] , [10, 2]]
amount = int(input('Ingrese la cantidad:'))
print(f"La cantidad seleccionada fue {amount}")
changes = coin_change2(coins, amount)
print(changes, len(changes))
```

UNIDAD 30 - QUIZ 2:

Dada la cantidad de monedas y la cantidad de cambio, encuentre la cantidad mínima de monedas que puede devolver como cambio.

```
import copy
def coin_change_pro(coins:list, amount:int): #guardando una copia de la
lista original
  #coins_first = coins.copy()
  coins_first = copy.deepcopy(coins)
  for low in range(len(coins)):
    changes = coin_change2(coins_first[low:], amount)
    if type(changes) != str:
      return changes, coins_first
    #reiniciando la wallet
    coins_first = copy.deepcopy(coins)
  return "No hay cambio disponible con las monedas del wallet.", coins

coins = [[500, 1] , [100, 1] , [50, 1 ] , [20, 3]]
amount = int(input('Ingrese la cantidad:'))
print(f"La cantidad seleccionada fue {amount}")
changes, coins = coin_change_pro(coins, amount)
print(changes, len(changes))
```

UNIDAD 31 - QUIZ 1:

Supongamos que hay ocho monedas idénticas numeradas del 1 al 8. De estas, solo una moneda es más pesada que las otra. Si puede pesar las monedas en una balanza de dos brazos, diseñe un algoritmo para seleccionar la moneda mas pesada.

Al menos, ¿cuántas veces se deben usar la balanza para encontrar la moneda?

```
def buscar_moneda_pesada(moneda_list):
  # dividir las monedas en dos grupos
  if len(moneda_list) != 1:
    mid = len(moneda_list) // 2
```

```
    grupo1 = moneda_list[:mid]
    grupo2 = moneda_list[mid:]
    # pesar los grupos
    peso = sum(grupo1) - sum(grupo2)
  else:
    #lista de un número, se encontró la moneda
    peso = 0
  # cuando el peso es cero, es que se encontró la moneda
  if peso == 0:
    return max(moneda_list), 0
  # si el grupo1 es más pesado, la moneda pesada está en ese grupo
  elif peso > 0:
    sub_moneda, sub_pesos = buscar_moneda_pesada(grupo1)
    return sub_moneda, sub_pesos + 1
  # si el grupo2 es más pesado, la moneda pesada está en ese grupo
  else:
    sub_moneda, sub_pesos = buscar_moneda_pesada(grupo2)
    return sub_moneda, sub_pesos + 1

buscar_moneda_pesada([2,2,2,2,2,10,2,2])
```

UNIDAD 31 - QUIZ 2:

En la pregunta anterior, supongamos que hay nueve monedas idénticas numeradas del 1 al 9. En este caso, diseñe un algoritmo para seleccionar una moneda pesada. Al menos, ¿cuántas veces se deben usar la balanza para sacar la moneda?

```
def buscar_moneda_pesada2(moneda_list):
  # dividir las monedas en dos grupos
  #__import__("ipdb").set_trace()
  if len(moneda_list) != 1:
    mid = len(moneda_list) // 3
    grupo1 = moneda_list[:mid]
    grupo2 = moneda_list[mid:2*mid]
    grupo3 = moneda_list[2*mid:]
    # pesar los grupos
    peso1 = sum(grupo1) - sum(grupo2)
    peso2 = sum(grupo2) - sum(grupo3)
    peso = -1
  else:
    #lista de un número, se encontró la moneda
    peso = 0
```

```
  # cuando el peso es cero, es que se encontró la moneda
  if peso == 0:
    return max(moneda_list), 0
  # si el grupo1 es más pesado, la moneda pesada está en ese grupo
  elif peso1 > 0:
    sub_moneda, sub_pesos = buscar_moneda_pesada2(grupo1)
    return sub_moneda, sub_pesos + 1
  # si el grupo2 es más pesado, la moneda pesada está en ese grupo
  elif peso2 < 0:
    sub_moneda, sub_pesos = buscar_moneda_pesada2(grupo3)
    return sub_moneda, sub_pesos + 1
  else:
    sub_moneda, sub_pesos = buscar_moneda_pesada2(grupo2)
    return sub_moneda, sub_pesos + 1

buscar_moneda_pesada2([2,2,2,2,2,10,2,2,2])
```

UNIDAD 31 - QUIZ 1:

```
def tromino_tile(board, size, missing_x, missing_y, start_x=0,
start_y=0):
    global TROMINO_TILE_COUNTER

    TROMINO_TILE_COUNTER += 1

    if size == 2:
        for i in range(start_x, start_x + size):
            for j in range(start_y, start_y + size):
                if i == missing_x and j == missing_y:
                    continue
                else:
                    board[i][j] = TROMINO_TILE_COUNTER
        return

    quarter_size = size // 2

    # Position of the missing tile in the four sub-quadrants
    missing_tile_quadrant = (
        missing_x // quarter_size,
        missing_y // quarter_size,
    )
```

```
    ### Tromino tile is placed in center of the board
    center_x = start_x + quarter_size
    center_y = start_y + quarter_size

    # Place tromino tile in the center of the board and recursively
call tromino_tile for each
    # of the four sub-quadrants
    for i in range(2):
        for j in range(2):
            # Calculating center moving around sub-quadrants
            dx = [-1, 0][i]
            dy = [-1, 0][j]

            if (i == missing_tile_quadrant[0]) & (j ==
missing_tile_quadrant[1]):
                # Running the same algorithm in the sub-quadrant with
missing tile
                tromino_tile(
                    board,
                    quarter_size,
                    missing_x,
                    missing_y,
                    center_x + (dx * quarter_size),
                    center_y + (dy * quarter_size),
                )
            else:
                # Placing tromino piece in board center
                board[center_x + dx][center_y + dy] =
TROMINO_TILE_COUNTER

                # Running the same algorithm in the sub-quadrant with
"placed tile"
                tromino_tile(
                    board,
                    quarter_size,
                    center_x + dx,
                    center_y + dy,
                    center_x + (dx * quarter_size),
                    center_y + (dy * quarter_size),
                )
import pandas as pd
# Example usage
```

```
TROMINO_TILE_COUNTER = 0
size = 8
missing_x = 2
missing_y = 1
board = [[0 for j in range(size)] for i in range(size)]
board[missing_x][missing_y] = -1
pd.DataFrame(board)

tromino_tile(board, size, missing_x, missing_y)

pd.DataFrame(board)
```

UNIDAD 32 - QUIZ 1:

Analice el resultado de la ejecución del siguiente código y compárelo con el rendimiento de las funciones *fib1()*, *fib2()* y *fib3()*.

```
inicio = time.time()
print((time.time() - inicio)*1000)
```

UNIDAD 32 - QUIZ 2:

Analice los resultados de ejecución del siguiente código y compárelo con el rendimiento de las funciones *bin1()* y *bin*

UNIDAD 32 - QUIZ 1:

Dados n enteros en un arreglo unidimensional sucesivo, escriba un algoritmo que encuentre cuándo se maximiza la suma de los valores sucesivos en el arreglo.

```
def max_sum_with_index(arr):
  n = len(arr)
  max_sum = float('-inf')
  current_sum = 0
  start_index = 0
  end_index = 0
  current_start_index = 0

  for i in range(n):
    current_sum += arr[i]
    #__import__("ipdb").set_trace()
    if current_sum > max_sum:
      max_sum = current_sum
```

```
      start_index = current_start_index
      end_index = i

    if current_sum < 0:
      current_sum = 0
      current_start_index = i+1

  return max_sum, start_index, end_index

S = [-2,1,-3,4,-1,2,1,-5,4]
M = max_sum_with_index(S)
print(M)
```

```
""" Ejercicio lógico: en cada palabra, reemplace las letras por un número, teniendo en cuenta que para cada palabra separada por un espacio la suma de sus dígitos es un número al cuadrado. Encuentra el número representado por cada letra. """ from math import sqrt, floor import time def is_square_digitsum(n): s = 0 #__import__("ipdb").set_trace() while n > 0: s += n % 10 n //= 10 if sqrt(s) == int(sqrt(s)): return True return False def find_all_squares(): sqrs = [[] for _ in range(5)] #__import__("ipdb").set_trace() for i in range(1, floor(sqrt(10 ** 5)) + 1): n = i * i if not is_square_digitsum(n): continue s = str(n) if len(s) == 3 and s[1] != s[2]: continue if len(s) == 5 and s[2] != s[3]: continue if len(s) in [4, 5] and len(set(s)) != 4: continue sqrs[len(s) - 1].append(n) return sqrs
def promising(s, n, dic): # s: palabra a verificar # n: numero a verificar # dic: diccionario letra-palabra #__import__("ipdb").set_trace() for i in range(len(s)): digit = int(str(n)[i]) for key, value in dic.items(): if key == s[i] and value != digit: return False if value == digit and key != s[i]: return False return True def solve(words, dic, squares): global solved if (len(words) == 0): solved = dic else: s = words[0] candidates = squares[len(s) - 1] #__import__("ipdb").set_trace() for n in candidates: if promising(s, n, dic): newdic = dic.copy() for i in range(len(s)): newdic[s[i]] = int(str(n)[i]) solve(words[1:], newdic, squares) def Main(): squares = find_all_squares() words = ['A', 'TO','ALL', 'XMAS', 'MERRY'] dic = {} solve(words, dic, squares) for word in words: print(word, end=": ") for c in word: print(solved[c], end="") print()
start = time.time() solved = {} Main() end = time.time() print("Tiempo transcurrido: ", end - start, " seconds")
```

```
def find_path(maze): n = len(maze) visited = [[False for _ in range(n)] for _ in range(n)] path = [] def dfs(x, y): visited[x][y] = True path.append((x, y)) __import__("ipdb").set_trace() if x ==
```

```
n - 1 and y == n - 1: return True for dx, dy in [(0, 1), (0, -1), (1, 0), (-1, 0)]: nx, ny = x + dx, y + dy if 0 <= nx < n and 0 <= ny < n and not visited[nx][ny] and maze[nx][ny] == 0: if dfs(nx, ny): return True path.pop() return False dfs(0, 0) return path if path else None
import numpy as np n = 5 # Tamaño del laberinto laberinto = np.random.randint(2, size=(n,n)) print(laberinto)
#laberinto[0][0] = 0 #laberinto[0][1] = 0 #laberinto[n-2][n-1] = 0 laberinto[n-1][n-1] = 0
find_path(laberinto)
```

UNIDAD 34 - QUIZ 1:

Seleccione aleatoriamente tres múltiplos de 5 en el rango de 0 a 100 utilizando el módulo **random** e imprímalos en forma de lista (escriba el código completo y la salida esperada).

```
import random
multiplos_5 = random.sample(range(0,100,5), 3)
print(multiplos_5)
```

UNIDAD 34 - QUIZ 2:

Use timedelta para crear un programa que imprima un aniversario de 100 días desde un día especial para usted. No tiene que ser un aniversario de 100 días, así que siéntase libre de hacer su propia calculadora de aniversario especial (escriba el código completo y la salida esperada).

```
import datetime
#from datetime import time

n = 100
d = datetime.datetime(2002, 5, 3)
delta = datetime.timedelta(days = n)

print(d + delta)
```

UNIDAD 34 - QUIZ 1:

El código de muestra a continuación ejecuta una obra de arte utilizando gráficos de Turtle. Turtle también es una de las bibliotecas estándar de Python y es muy fácil de usar. Las descripciones del módulo de gráficos Python Turtle se encuentran en el siguiente enlace:

https://docs.python.org/3/library/turtle.html?highlight=turtle#module-turtle.

```
# Hacer un patrón de arcoiris geométrico
import turtle
colors=['red','yellow','blue','orange','green','red']
```

```
aiden = turtle.Turtle()
turtle.bgcolor('black') # Convierte el fondo negro

# Hacer 36 hexágonos, cada uno de 10° de separación
for n in range(36):
# Hacer el hexágono repitiendo este proceso 6 veces
  for i in range(6):
    aiden.color(colors[i]) # Elegir el color en la posición i
    aiden.forward(100)
    aiden.left(60)
    # Añadir un giro de 10° antes del siguiente hexágono
    aiden.right(10)

# Prepárate para dibujar 36 círculos
aiden.penup()
aiden.color('white')
# Repita 36 veces para unir los 36 hexágonos
for i in range(36):
  aiden.forward(220)
  aiden.pendown()
  aiden.circle(5)
  aiden.penup()
  aiden.backward(220)
  aiden.right(10)
# Ocultar Turtle para terminar el dibujo
aiden.hideturtle()
```

UNIDAD 35 - QUIZ 1:

El conjunto de datos de Spotify tiene varios valores de columna como el nombre del artista, la clasificación y la popularidad que usamos en la misión. Hable con su compañero para crear una lista de reproducción especial que se haga con otros valores de columna.

```
from google.colab import files
files.upload()

tracks = pd.read_csv("tracks_features.csv")
tracks.head()
```

```
tracks.columns
tracks.info()
tracks.shape
tracks["year"].value_counts()
playlist_1 = tracks[(tracks["explicit"] == False) &
(tracks["danceability"] >= 0.7) & (tracks["year"] == 2020)]
```

UNIDAD 36 - QUIZ 1:

En el conjunto de datos de Spotify, hay varios valores de columna además del nombre, el rango y la popularidad del artista que usamos en nuestra misión. Hable con sus colegas de aprendizaje para crear listas de reproducción especiales usando diferentes valores de columna.

Ejemplo: Una lista de reproducción de solo música alegre/animada: use la columna "tempo" para seleccionar un tempo apropiado.

TIP

Si ha creado una lista de reproducción, guárdela como un archivo de Excel y compártala con sus colegas de aprendizaje. Guardar un DataFrame como un archivo de Excel es muy sencillo.

pandas.DataFrame.to_excel()

Para mayor información:

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_excel.html

```
from google.colab import drive
drive.mount('/content/gdrive')

tracks = pd.read_csv("/content/gdrive/MyDrive/tracks_features.csv")
tracks.head()

tracks.describe()

tracks.columns

playlist_2 = tracks[(tracks["explicit"] == False) &
(tracks["danceability"] >= 0.8) & (tracks["tempo"] >= 150) &
(tracks['year'] >= 2018) & (tracks["valence"] > 0.95)]
```

```
playlist_2

ruta = '/content/gdrive/My Drive/Ejemplo.xlsx'
playlist_2.to_excel(ruta, index=False)
```