### Introducción a Python

**Ejercicio (¿Qué es Python?)**

Ejecute el siguiente código y vea qué sucede. Siéntase libre de cambiarlo y ejecutarlo tantas veces como desee.

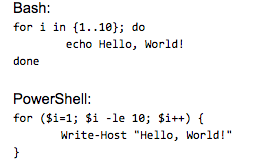
friends = ['Taylor', 'Alex', 'Pat', 'Eli']

for friend in friends:

    print("Hi " + friend)

#### Ejercicio (Otros lenguajes)

Así es como se ve imprimir "Hola, mundo" 10 veces en Bash y Powershell:



Ahora pruebe el ejemplo de Python usted mismo:

for i in range(10):

  print("Hello, World!")

### Hola Mundo

#### Ejercicio (¡Hola Mundo!)

Ahora que sabe lo que está pasando con este fragmento de código, intente hacer que Python genere una cadena diferente. Escriba una secuencia de comandos de Python que dé como resultado "Estoy programando en Python". a la pantalla. Recuerde que necesita usar la función print() y usar comillas para delimitar la cadena.

print("I'm programming in Python!")

#### Ejercicio (Obtener información del usuario)

¡Pruébalo tú mismo! En el siguiente script, cambie los valores de color y cosa para que la computadora genere una declaración diferente a la inicial.

color = "Green"

thing = "Hope"

print(color + " is the color of " + thing)

color = "Red"

thing = "Heart"

print(color + " is the color of " + thing)

#### Ejercicio (Python puede ser tu calculadora)

¡Ahora es tu turno de intentarlo! Pruébalo tú mismo.

Usa Python para calcular (((1+2)\*3)/4)^5(((1+2)∗3)/4)5

**Consejo** : recuerde que puede usar a \*\* b para calcular a elevado a b.

print()

print((((1+2)\*3)/4)\*\*5)

#### Prueba de práctica: Hola mundo

Genere un mensaje que diga "¡Programar en Python es divertido!" a la pantalla.

print("Programming in Python is fun!")

Reemplace el marcador de posición \_\_\_ y ​​calcule la proporción áurea: \ frac {1+ \ sqrt {5}} {2}21+5​​.

Consejo: para calcular la raíz cuadrada de un número x*x*, puede utilizar x \*\* (1/2).

ratio = (1+(5\*\*(1/2)))/2

print(ratio)

### Expresiones y variables

#### Ejercicio (Tipos de datos)

¿Por qué este código genera un error?

print("1234"+5678)

Porque los números no deben escribirse entre comillas.

#### Ejercicio (Variables)

¡Ahora es tu turno de intentarlo!

Complete los espacios en blanco para calcular el área de un triángulo de base 5, altura 3 y envíe el resultado. **Recordatorio:** el área de un triángulo es (base \* altura) / 2.

base = \_\_\_

height = \_\_\_

area = \_\_\_

print(area)

base = 5

height = 3

area = (base \* height) / 2

print(area)

#### Ejercicio (Conversiones de expresiones, números y tipos)

Practica escribiendo algunas expresiones y conversiones tú mismo.

En este escenario, tenemos un directorio con 5 archivos. Cada archivo tiene un tamaño diferente: 2048, 4357, 97658, 125 y 8. Complete los espacios en blanco para calcular el tamaño promedio del archivo haciendo que Python agregue todos los valores por usted y luego establezca la variable de archivos en el número de archivos. Por último, envíe un mensaje que diga "El tamaño medio es:" seguido del número resultante. Recuerde usar la función str () para convertir el número en una cadena.

total = 2048 + \_\_\_ + \_\_\_ + \_\_\_ + \_\_\_

files = \_\_\_

average = total / files

print("\_\_\_" + str(\_\_\_))

total = 2048 + 4357 + 97658 + 125 + 8

files = 5

average = total / files

print("The average size is: " + str(average))

**Prueba de práctica: Expresiones y variables**

Pregunta 1

En este escenario, dos amigos están cenando en un restaurante. La factura viene por la cantidad de 47,28 dólares. Los amigos deciden dividir la cuenta en partes iguales, luego de agregar un 15% de propina por el servicio. Calcule la propina, la cantidad total a pagar y la parte de cada amigo, luego envíe un mensaje que diga "Cada persona debe pagar:" seguido del número resultante.

bill = \_\_\_

tip = bill \* \_\_\_

total = bill + \_\_\_

share = \_\_\_

print("")

bill = 47.28

tip = bill \* 0.15

total = bill + tip

share = total / 2

print("Each person needs to pay: " + str(share))

Pregunta 2

Se supone que este código toma dos números, divide uno por otro para que el resultado sea igual a 1 y muestra el resultado en la pantalla. Desafortunadamente, hay un error en el código. Encuentre el error y corríjalo para que el resultado sea correcto.

numerator = 10

denominator = 0

result = numerator / denominator

print(result)

numerator = 10

denominator = 10

result = numerator / denominator

print(result)

Pregunta 3

Combine las variables para mostrar la oración "¿Qué le parece Python hasta ahora?"

word1 = "How"

word2 = "do"

word3 = "you"

word4 = "like"

word5 = "Python"

word6 = "so"

word7 = "far?"

print(\_\_\_)

word1 = "How"

word2 = "do"

word3 = "you"

word4 = "like"

word5 = "Python"

word6 = "so"

word7 = "far?"

print(word1 + " " + word2 + " " + word3 + " " + word4 + " " + word5 + " " + word6 + " " +  word7)

Pregunta 4

Se supone que este código muestra "2 + 2 = 4" en la pantalla, pero hay un error. Encuentre el error en el código y corríjalo, para que el resultado sea correcto.

print("2 + 2 = " + (2 + 2))

print("2 + 2 = " + str(2 + 2))

### Funciones

#### Ejercicio (Definición de funciones)

¿Crees que puedes desarrollar tu propia función? ¡Yo creo que puedes! Vamos a intentarlo.

Amplíe el cuerpo de la función print\_seconds para que imprima la cantidad total de segundos dados los parámetros de la función horas, minutos y segundos. Recuerde que hay 3600 segundos en una hora y 60 segundos en un minuto.

def print\_seconds(hours, minutes, seconds):

    print(\_\_\_)

print\_seconds(1,2,3)

def print\_seconds(hours, minutes, seconds):

    print((3600 \* hours) + (60 \* minutes) + seconds)

print\_seconds(1,2,3)

#### Ejercicio (Devolución de valores)

Utilice la función get\_seconds para calcular la cantidad de segundos en 2 horas y 30 minutos, luego agregue este número a la cantidad de segundos en 45 minutos y 15 segundos. Luego imprime el resultado.

def get\_seconds(hours, minutes, seconds):

  return 3600\*hours + 60\*minutes + seconds

amount\_a = get\_seconds(\_\_\_)

amount\_b = get\_seconds(\_\_\_)

result = \_\_\_

print(result)

def get\_seconds(hours, minutes, seconds):

  return 3600\*hours + 60\*minutes + seconds

amount\_a = get\_seconds(2,30,0)

amount\_b = get\_seconds(0,45,15)

result = amount\_a + amount\_b

print(result)

#### Ejercicio (Los principios de la reutilización del código)

¿Listo para probarlo tú mismo? Vea si puede reducir la duplicación de código en este script.

En este código, identifique el patrón repetido y reemplácelo con una función llamada month\_days, que recibe el nombre del mes y la cantidad de días en ese mes como parámetros. Adapte el resto del código para que el resultado sea el mismo. Confirme sus resultados haciendo una llamada de función con los parámetros correctos para ambos meses enumerados.

# REPLACE THIS STARTER CODE WITH YOUR FUNCTION

june\_days = 30

print("June has " + str(june\_days) + " days.")

july\_days = 31

print("July has " + str(july\_days) + " days.")

def month\_days(month, days):

    print(month + " has " + str(days) + " days.")

month\_days("June", 30)

month\_days("July", 31)

#### Ejercicio (Estilo de código)

Esta función para calcular el área de un rectángulo no es muy legible. ¿Puedes refactorizarlo y luego llamar a la función para calcular el área con base de 5 y altura de 6? **Sugerencia:** una función que calcula el área de un rectángulo probablemente debería llamarse rectangle\_area, y si recibe base y altura, así es como deberían llamarse los parámetros.

def f1(x, y):

    z = x\*y  # the area is base\*height

    print("The area is " + str(z))

def rectangle\_area(base, height):

    area = base\*height  # the area is base\*height

    print("The area is " + str(area))

rectangle\_area(5, 6)

#### Prueba de práctica: funciones

Pregunta 1

Esta función convierte millas a kilómetros (km).

1. Completa la función para devolver el resultado de la conversión.
2. Llame a la función para convertir la distancia del viaje de millas a kilómetros
3. Complete el espacio en blanco para imprimir el resultado de la conversión
4. Calcule el viaje de ida y vuelta en kilómetros duplicando el resultado y complete el espacio en blanco para imprimir el resultado.

# 1) Complete the function to return the result of the conversion

def convert\_distance(miles):

    km = miles \* 1.6  # approximately 1.6 km in 1 mile

my\_trip\_miles = 55

# 2) Convert my\_trip\_miles to kilometers by calling the function above

my\_trip\_km = \_\_\_

# 3) Fill in the blank to print the result of the conversion

print("The distance in kilometers is " + \_\_\_)

# 4) Calculate the round-trip in kilometers by doubling the result,

#    and fill in the blank to print the result

print("The round-trip in kilometers is " + \_\_\_)

# 1) Complete the function to return the result of the conversion

def convert\_distance(miles):

    km = miles \* 1.6  # approximately 1.6 km in 1 mile

    return km

my\_trip\_miles = 55

# 2) Convert my\_trip\_miles to kilometers by calling the function above

my\_trip\_km = convert\_distance(my\_trip\_miles)

# 3) Fill in the blank to print the result of the conversion

print("The distance in kilometers is " + str(my\_trip\_km))

# 4) Calculate the round-trip in kilometers by doubling the result,

#    and fill in the blank to print the result

print("The round-trip in kilometers is " + str(my\_trip\_km\*2))

Pregunta 2

Esta función compara dos números y los devuelve en orden creciente.

1. Complete los espacios en blanco para que la declaración de impresión muestre el resultado de la llamada a la función en orden.

Sugerencia: si una función devuelve varios valores, no olvide almacenar estos valores en varias variables

# This function compares two numbers and returns them

# in increasing order.

def order\_numbers(number1, number2):

    if number2 > number1:

        return number1, number2

    else:

        return number2, number1

# 1) Fill in the blanks so the print statement displays the result

#    of the function call

\_\_\_, \_\_\_ = order\_numbers(100, 99)

print(smaller, bigger)

# This function compares two numbers and returns them

# in increasing order.

def order\_numbers(number1, number2):

    if number2 > number1:

        return number1, number2

    else:

        return number2, number1

# 1) Fill in the blanks so the print statement displays the result

#    of the function call

smaller, bigger = order\_numbers(100, 99)

print(smaller, bigger)

Pregunta 3

Revisemos nuestra función lucky\_number. Queremos cambiarlo, para que en lugar de imprimir el mensaje, devuelva el mensaje. De esta manera, la línea que llama puede imprimir el mensaje o hacer algo más con él si es necesario. Complete los espacios en blanco para completar el código para que funcione.

def lucky\_number(name):

  number = len(name) \* 9

  \_\_\_ = "Hello " + name + ". Your lucky number is " + str(number)

  \_\_\_

print(lucky\_number("Kay"))

print(lucky\_number("Cameron"))

def lucky\_number(name):

  number = len(name) \* 9

  message = "Hello " + name + ". Your lucky number is " + str(number)

  return message

print(lucky\_number("Kay"))

print(lucky\_number("Cameron"))

### Condicionales

#### Ejercicio (Comparando cosas)

Averigüe cuál es la relación entre las cadenas "cat" y "Cat" reemplazando el signo más con operadores de comparación.

print("cat" > "Cat")

#### Ejercicio (Ramificación con declaraciones if)

La función is\_positive debería devolver True si el número recibido es positivo; de lo contrario, devuelve None. ¿Puede llenar los vacíos para que eso suceda?

def is\_positive(number):

  if \_\_\_:

    return \_\_\_

def is\_positive(number):

  if number > 0:

    return True

#### Ejercicio (Declaraciones else)

La función is\_positive debería devolver True si el número recibido es positivo y False si no lo es. ¿Puede llenar los vacíos para que eso suceda?

def is\_positive(number):

  if number > 0:

    return True

  else:

    \_\_\_

def is\_positive(number):

  if number > 0:

    return True

  else:

    return False

#### Ejercicio (Declaraciones elif)

La función number\_group debe devolver "Positivo" si el número recibido es positivo, "Negativo" si es negativo y "Cero" si es 0. ¿Puede llenar los espacios para que eso suceda?

def number\_group(number):

  if \_\_\_:

    return "Positive"

  elif \_\_\_:

    return \_\_\_

  else:

    \_\_\_

print(number\_group(10)) #Should be Positive

print(number\_group(0)) #Should be Zero

print(number\_group(-5)) #Should be Negative

def number\_group(number):

  if number > 0:

    return "Positive"

  elif number < 0:

    return "Negative"

  else:

    return "Zero"

print(number\_group(10)) #Should be Positive

print(number\_group(0)) #Should be Zero

print(number\_group(-5)) #Should be Negative

#### Prueba de práctica: condicionales

Pregunta 1

Complete el guión completando las partes que faltan. La función recibe un nombre y luego devuelve un saludo en función de si ese nombre es "Taylor" o no.

def greeting(name):

  if \_\_\_ == "Taylor":

    return "Welcome back Taylor!"

  \_\_\_:

    return "Hello there, " + name

print(greeting("Taylor"))

print(greeting("John"))

def greeting(name):

  if name == "Taylor":

    return "Welcome back Taylor!"

  else:

    return "Hello there, " + name

print(greeting("Taylor"))

print(greeting("John"))

Pregunta 2

¿Cuál es el resultado de este código si el número es igual a 10?

if number > 11:

  print(0)

elif number != 10:

  print(1)

elif number >= 20 or number < 12:

  print(2)

else:

  print(3)

Pregunta 3

Si un sistema de archivos tiene un tamaño de bloque de 4096 bytes, esto significa que un archivo compuesto por solo un byte seguirá usando 4096 bytes de almacenamiento. Un archivo compuesto por 4097 bytes utilizará 4096 \* 2 = 8192 bytes de almacenamiento. Sabiendo esto, ¿puede completar los huecos en la función calculate\_storage a continuación, que calcula el número total de bytes necesarios para almacenar un archivo de un tamaño determinado?

def calculate\_storage(filesize):

    block\_size = 4096

    # Use floor division to calculate how many blocks are fully occupied

    full\_blocks = \_\_\_

    # Use the modulo operator to check whether there's any remainder

    partial\_block\_remainder = \_\_\_

    # Depending on whether there's a remainder or not, return

    # the total number of bytes required to allocate enough blocks

    # to store your data.

    if partial\_block\_remainder > 0:

        return \_\_\_

    return \_\_\_

print(calculate\_storage(1))    # Should be 4096

print(calculate\_storage(4096)) # Should be 4096

print(calculate\_storage(4097)) # Should be 8192

print(calculate\_storage(6000)) # Should be 8192

def calculate\_storage(filesize):

    block\_size = 4096

    # Use floor division to calculate how many blocks are fully occupied

    full\_blocks = filesize // block\_size

    # Use the modulo operator to check whether there's any remainder

    partial\_block\_remainder = filesize % block\_size

    # Depending on whether there's a remainder or not, return

    # the total number of bytes required to allocate enough blocks

    # to store your data.

    if partial\_block\_remainder > 0:

        return (full\_blocks + 1)\*block\_size

    return full\_blocks\* block\_size

print(calculate\_storage(1))    # Should be 4096

print(calculate\_storage(4096)) # Should be 4096

print(calculate\_storage(4097)) # Should be 8192

print(calculate\_storage(6000)) # Should be 8192

#### Evaluación calificada del módulo 2

Pregunta 1

Complete la función completando las partes que faltan. La función color\_translator recibe el nombre de un color y luego imprime su valor hexadecimal. Actualmente, solo admite los tres colores primarios aditivos (rojo, verde, azul), por lo que devuelve "desconocido" para todos los demás colores.

def color\_translator(color):

    if \_\_\_ == "red":

        hex\_color = "#ff0000"

    elif \_\_\_ == "green":

        hex\_color = "#00ff00"

    elif \_\_\_ == "blue":

        hex\_color = "#0000ff"

    \_\_\_:

        hex\_color = "unknown"

    return \_\_\_

print(color\_translator("blue")) # Should be #0000ff

print(color\_translator("yellow")) # Should be unknown

print(color\_translator("red")) # Should be #ff0000

print(color\_translator("black")) # Should be unknown

print(color\_translator("green")) # Should be #00ff00

print(color\_translator("")) # Should be unknown

def color\_translator(color):

    if color == "red":

        hex\_color = "#ff0000"

    elif color == "green":

        hex\_color = "#00ff00"

    elif color == "blue":

        hex\_color = "#0000ff"

    else:

        hex\_color = "unknown"

    return hex\_color

print(color\_translator("blue")) # Should be #0000ff

print(color\_translator("yellow")) # Should be unknown

print(color\_translator("red")) # Should be #ff0000

print(color\_translator("black")) # Should be unknown

print(color\_translator("green")) # Should be #00ff00

print(color\_translator("")) # Should be unknown

Pregunta 4

Los estudiantes de una clase reciben sus calificaciones como aprobado / reprobado. Las puntuaciones de 60 o más (de 100) significan que la calificación es "Aprobado". Para puntuaciones más bajas, la calificación es "Reprobar". Además, las puntuaciones superiores a 95 (no incluidas) se califican como "Puntuación máxima". Complete esta función para que devuelva la calificación adecuada.

def exam\_grade(score):

    if \_\_\_:

        grade = "Top Score"

    elif \_\_\_:

        grade = "Pass"

    else:

        grade = "Fail"

    return grade

print(exam\_grade(65)) # Should be Pass

print(exam\_grade(55)) # Should be Fail

print(exam\_grade(60)) # Should be Pass

print(exam\_grade(95)) # Should be Pass

print(exam\_grade(100)) # Should be Top Score

print(exam\_grade(0)) # Should be Fail

def exam\_grade(score):

    if score > 95:

        grade = "Top Score"

    elif score >= 60:

        grade = "Pass"

    else:

        grade = "Fail"

    return grade

print(exam\_grade(65)) # Should be Pass

print(exam\_grade(55)) # Should be Fail

print(exam\_grade(60)) # Should be Pass

print(exam\_grade(95)) # Should be Pass

print(exam\_grade(100)) # Should be Top Score

print(exam\_grade(0)) # Should be Fail

Pregunta 6

Complete el cuerpo de la función ***format\_name*** . Esta función recibe los parámetros ***first\_name*** y ***last\_name*** y luego devuelve una cadena con el formato adecuado.

Específicamente:

Si se ***proporcionan*** los parámetros ***last\_name*** y ***first\_name*** , la función debería regresar así:

Si solo se proporciona *un* parámetro de nombre (ya sea el nombre *o* el apellido), la función debería regresar así:

print(format\_name("Ella", "Fitzgerald"))

Name: Fitzgerald, Ella

Si solo se proporciona **un** parámetro de nombre (ya sea el nombre o el apellido), la función debería regresar así:

Name: Adele

O:

print(format\_name("", "Einstein"))

Name: Einstein

Finalmente, si ambos nombres están en blanco, la función debería devolver la cadena vacía:

print(format\_name("", ""))

Implementar a continuación:

def format\_name(first\_name, last\_name):

    # code goes here

    return string

print(format\_name("Ernest", "Hemingway"))

# Should return the string "Name: Hemingway, Ernest"

print(format\_name("", "Madonna"))

# Should return the string "Name: Madonna"

print(format\_name("Voltaire", ""))

# Should return the string "Name: Voltaire"

print(format\_name("", ""))

# Should return an empty string

def format\_name(first\_name, last\_name):

    # code goes here

    if (first\_name != "") and (last\_name != ""):

        string = "Name: " + last\_name + ", " + first\_name

    elif (first\_name == "") and (last\_name == ""):

        string = ""

    else:

        string = "Name: " + first\_name + last\_name

    return string

print(format\_name("Ernest", "Hemingway"))

# Should return the string "Name: Hemingway, Ernest"

print(format\_name("", "Madonna"))

# Should return the string "Name: Madonna"

print(format\_name("Voltaire", ""))

# Should return the string "Name: Voltaire"

print(format\_name("", ""))

# Should return an empty string

Pregunta 7

La función longgest\_word se utiliza para comparar 3 palabras. Debe devolver la palabra con la mayor cantidad de caracteres (y la primera en la lista cuando tienen la misma longitud). Complete el espacio en blanco para que esto suceda.

def longest\_word(word1, word2, word3):

    if len(word1) >= len(word2) and len(word1) >= len(word3):

        word = word1

    elif \_\_\_:

        word = word2

    else:

        word = word3

    return(word)

print(longest\_word("chair", "couch", "table"))

print(longest\_word("bed", "bath", "beyond"))

print(longest\_word("laptop", "notebook", "desktop"))

def longest\_word(word1, word2, word3):

    if len(word1) >= len(word2) and len(word1) >= len(word3):

        word = word1

    elif len(word2) >= len(word1) and len(word2) >= len(word3):

        word = word2

    else:

        word = word3

    return(word)

print(longest\_word("chair", "couch", "table"))

print(longest\_word("bed", "bath", "beyond"))

print(longest\_word("laptop", "notebook", "desktop"))

Pregunta 10

La función fracción\_fraccional divide el numerador por el denominador y devuelve solo la parte fraccionaria (un número entre 0 y 1). Completa el cuerpo de la función para que devuelva el número correcto. Nota: Dado que la división por 0 produce un error, si el denominador es 0, la función debe devolver 0 en lugar de intentar la división.

def fractional\_part(numerator, denominator):

    # Operate with numerator and denominator to

# keep just the fractional part of the quotient

    return 0

print(fractional\_part(5, 5)) # Should be 0

print(fractional\_part(5, 4)) # Should be 0.25

print(fractional\_part(5, 3)) # Should be 0.66...

print(fractional\_part(5, 2)) # Should be 0.5

print(fractional\_part(5, 0)) # Should be 0

print(fractional\_part(0, 5)) # Should be 0

def fractional\_part(numerator, denominator):

    # Operate with numerator and denominator to

    if denominator > 0:

        return (numerator / denominator) - (numerator // denominator)

# keep just the fractional part of the quotient

    return 0

print(fractional\_part(5, 5)) # Should be 0

print(fractional\_part(5, 4)) # Should be 0.25

print(fractional\_part(5, 3)) # Should be 0.66...

print(fractional\_part(5, 2)) # Should be 0.5

print(fractional\_part(5, 0)) # Should be 0

print(fractional\_part(0, 5)) # Should be 0

### While Loops

#### Ejercicio (Por qué es importante inicializar variables)

En este código, hay un problema de inicialización que hace que nuestra función se comporte incorrectamente. ¿Puedes encontrar el problema y solucionarlo?

def count\_down(start\_number):

  while (current > 0):

    print(current)

    current -= 1

  print("Zero!")

count\_down(3)

def count\_down(start\_number):

  current = start\_number

  while (current > 0):

    print(current)

    current -= 1

  print("Zero!")

count\_down(3)

#### Ejercicio (Bucles infinitos y cómo romperlos)

El siguiente código provoca un bucle infinito. ¿Puedes averiguar qué falta y cómo solucionarlo?

def print\_range(start, end):

    # Loop through the numbers from start to end

    n = start

    while n <= end:

        print(n)

print\_range(1, 5)  # Should print 1 2 3 4 5 (each number on its own line)

def print\_range(start, end):

    # Loop through the numbers from start to end

    n = start

    while n <= end:

        print(n)

        n += 1

print\_range(1, 5)  # Should print 1 2 3 4 5 (each number on its own line)

#### Prueba de práctica: bucles while

Pregunta 1

Complete los espacios en blanco para que la función print\_prime\_factors imprima todos los factores primos de un número. Un factor primo es un número que es primo y divide a otro sin dejar residuo.

def print\_prime\_factors(number):

  # Start with two, which is the first prime

  factor = \_\_\_

  # Keep going until the factor is larger than the number

  while factor <= number:

    # Check if factor is a divisor of number

    if number % factor == \_\_\_:

      # If it is, print it and divide the original number

      print(factor)

      number = number / factor

    else:

      # If it's not, increment the factor by one

      \_\_\_

  return "Done"

print\_prime\_factors(100)

# Should print 2,2,5,5

# DO NOT DELETE THIS COMMENT

def print\_prime\_factors(number):

  # Start with two, which is the first prime

  factor = 2

  # Keep going until the factor is larger than the number

  while factor <= number:

    # Check if factor is a divisor of number

    if number % factor == 0:

      # If it is, print it and divide the original number

      print(factor)

      number = number / factor

    else:

      # If it's not, increment the factor by one

      factor += 1

  return "Done"

print\_prime\_factors(100)

# Should print 2,2,5,5

# DO NOT DELETE THIS COMMENT

Pregunta 2

El siguiente código puede conducir a un bucle infinito. Corrija el código para que pueda finalizar correctamente para todos los números.

Nota: Intente ejecutar su función con el número 0 como entrada, ¡y vea lo que obtiene!

def is\_power\_of\_two(n):

  # Check if the number can be divided by two without a remainder

  while n % 2 == 0:

    n = n / 2

  # If after dividing by two the number is 1, it's a power of two

  if n == 1:

    return True

  return False

print(is\_power\_of\_two(0)) # Should be False

print(is\_power\_of\_two(1)) # Should be True

print(is\_power\_of\_two(8)) # Should be True

print(is\_power\_of\_two(9)) # Should be False

def is\_power\_of\_two(n):

  # Check if the number can be divided by two without a remainder

  while n > 0 and n % 2 == 0:

    n = n / 2

  # If after dividing by two the number is 1, it's a power of two

  if n == 1:

    return True

  return False

print(is\_power\_of\_two(0)) # Should be False

print(is\_power\_of\_two(1)) # Should be True

print(is\_power\_of\_two(8)) # Should be True

print(is\_power\_of\_two(9)) # Should be False

Pregunta 3

Complete la función vacía para que devuelva la suma de todos los divisores de un número, sin incluirlo. Un divisor es un número que se divide en otro sin dejar residuo.

def sum\_divisors(n):

  sum = 0

  # Return the sum of all divisors of n, not including n

  return sum

print(sum\_divisors(0))

# 0

print(sum\_divisors(3)) # Should sum of 1

# 1

print(sum\_divisors(36)) # Should sum of 1+2+3+4+6+9+12+18

# 55

print(sum\_divisors(102)) # Should be sum of 2+3+6+17+34+51

# 114

def sum\_divisors(n):

  sum = 0

  x = 1

  # Return the sum of all divisors of n, not including n

  while x < n:

    if n % x == 0:

      sum += x

    x += 1

  return sum

print(sum\_divisors(0))

# 0

print(sum\_divisors(3)) # Should sum of 1

# 1

print(sum\_divisors(36)) # Should sum of 1+2+3+4+6+9+12+18

# 55

print(sum\_divisors(102)) # Should be sum of 2+3+6+17+34+51

# 114

Pregunta 4

La función multiplication\_table imprime los resultados de un número que se le pasa multiplicado por 1 a 5. Un requisito adicional es que el resultado no exceda 25, lo cual se hace con la instrucción break. Complete los espacios en blanco para completar la función para satisfacer estas condiciones.

def multiplication\_table(number):

    # Initialize the starting point of the multiplication table

    multiplier = 1

    # Only want to loop through 5

    while multiplier <= 5:

        result = \_\_\_

        # What is the additional condition to exit out of the loop?

        if \_\_\_ :

            break

        print(str(number) + "x" + str(multiplier) + "=" + str(result))

        # Increment the variable for the loop

        \_\_\_ += 1

multiplication\_table(3)

# Should print: 3x1=3 3x2=6 3x3=9 3x4=12 3x5=15

multiplication\_table(5)

# Should print: 5x1=5 5x2=10 5x3=15 5x4=20 5x5=25

multiplication\_table(8)

# Should print: 8x1=8 8x2=16 8x3=24

def multiplication\_table(number):

    # Initialize the starting point of the multiplication table

    multiplier = 1

    # Only want to loop through 5

    while multiplier <= 5:

        result = number \* multiplier

        # What is the additional condition to exit out of the loop?

        if result > 25 :

            break

        print(str(number) + "x" + str(multiplier) + "=" + str(result))

        # Increment the variable for the loop

        multiplier += 1

multiplication\_table(3)

# Should print: 3x1=3 3x2=6 3x3=9 3x4=12 3x5=15

multiplication\_table(5)

# Should print: 5x1=5 5x2=10 5x3=15 5x4=20 5x5=25

multiplication\_table(8)

# Should print: 8x1=8 8x2=16 8x3=24

### For Loops

#### Ejercicio (¿Qué es un bucle for?)

¿Quieres probar esto? ¡Vamos a intentarlo!

Complete los espacios en blanco de la función sum\_squares, de modo que devuelva la suma de todos los cuadrados de números entre 0 y x (no incluidos). Recuerde que puede usar la función range (x) para generar una secuencia de números del 0 al x (no incluido).

def square(n):

    return n\*n

def sum\_squares(x):

    sum = 0

    for n in \_\_\_:

        sum += \_\_

    return \_\_

print(sum\_squares(10)) # Should be 285

def square(n):

    return n\*n

def sum\_squares(x):

    sum = 0

    for n in range(x):

        sum += square(n)

    return sum

print(sum\_squares(10)) # Should be 285

#### Ejercicio Más para ejemplos de bucles)

En matemáticas, el factorial de un número se define como el producto de un número entero y todos los números enteros por debajo de él. Por ejemplo, el factorial de cuatro (¡4!) Es igual a 1 \* 2 \* 3 \* 4 = 24. Complete los espacios en blanco para que la función factorial devuelva el número correcto.

def factorial(n):

    result = 1

    for i in \_\_\_:

        \_\_

    return result

print(factorial(4)) # should return 24

print(factorial(5)) # should return 120

def factorial(n):

    result = 1

    for i in range(1,n):

        result += result \* i

    return result

print(factorial(4)) # should return 24

print(factorial(5)) # should return 120

#### Ejercicio (Anidado para bucles)

Dado el siguiente código:

teams = [ 'Dragons', 'Wolves', 'Pandas', 'Unicorns']

for home\_team in teams:

    for away\_team in teams:

¿Cuál debería ser la siguiente línea para evitar que ambas variables se impriman con el mismo valor?

* while home\_team! = away\_team:
* for home\_team == away\_team:
* away\_team = home\_team
* if home\_team != away\_team: <<<<<<<

#### Ejercicio (Errores comunes en bucles for)

El sistema utiliza la función validate\_users para comprobar si una lista de usuarios es válida o no. Un usuario válido es aquel que tiene al menos 3 caracteres. Por ejemplo, ['taylor', 'luisa', 'jamaal'] son ​​todos usuarios válidos. Al llamarlo como en este ejemplo, algo no está bien. ¿Puedes averiguar qué arreglar?

def validate\_users(users):

  for user in users:

    if is\_valid(user):

      print(user + " is valid")

    else:

      print(user + " is invalid")

validate\_users("purplecat")

def validate\_users(users):

  for user in users:

    if is\_valid(user):

      print(user + " is valid")

    else:

      print(user + " is invalid")

validate\_users(["purplecat"])

#### Prueba de práctica: bucles for

Pregunta 1

Complete los espacios en blanco para que la función factorial devuelva el factorial de n. Luego, imprime los primeros 10 factoriales (del 0 al 9) con el número correspondiente. Recuerda que el factorial de un número se define como el producto de un número entero y todos los números enteros anteriores. Por ejemplo, el factorial de cinco (¡5!) Es igual a 1 \* 2 \* 3 \* 4 \* 5 = 120. Recuerde también que el factorial de cero (¡0!) Es igual a 1.

def factorial(n):

    result = 1

    for x in range(1,\_\_\_):

        result = \_\_\_ \* \_\_\_

    return \_\_\_

for n in range(\_\_\_,\_\_\_):

    print(n, factorial(n+\_\_\_))

def factorial(n):

    result = 1

    for x in range(1,n):

        result += result \* x

    return result

for n in range(0,10):

    print(n, factorial(n))

Pregunta 2

Escribe un guión que imprima los primeros 10 números de cubo (x \*\* 3), comenzando con x = 1 y terminando con x = 10.

for \_\_ in range(\_\_,\_\_):

  print(\_\_)

for x in range(1,11):

  print(str(x\*\*3))

Pregunta 3

Escriba un guión que imprima los múltiplos de 7 entre 0 y 100. Imprima un múltiplo por línea y evite imprimir números que no sean múltiplos de 7. Recuerde que 0 también es múltiplo de 7.

print()

for x in range(101):

    if (x\*7) < 100:

        print(x\*7)

Pregunta 4

La función de reintento intenta ejecutar una operación que podría fallar, vuelve a intentar la operación durante varios intentos. Actualmente, el código seguirá ejecutando la función incluso si tiene éxito. Complete el espacio en blanco para que el código deje de intentarlo después de que la operación se haya realizado correctamente.

def retry(operation, attempts):

  for n in range(attempts):

    if operation():

      print("Attempt " + str(n) + " succeeded")

    \_\_\_

    else:

      print("Attempt " + str(n) + " failed")

retry(create\_user, 3)

retry(stop\_service, 5)

def retry(operation, attempts):

  for n in range(attempts):

    if operation():

      print("Attempt " + str(n) + " succeeded")

      break

    else:

      print("Attempt " + str(n) + " failed")

retry(create\_user, 3)

retry(stop\_service, 5)

### Recursion

#### Ejercicio (¿Qué es la recursividad?)

La función sum\_positive\_numbers debería devolver la suma de todos los números positivos entre el número n recibido y 1. Por ejemplo, cuando n es 3 debería devolver 1 + 2 + 3 = 6, y cuando n es 5 debería devolver 1 + 2 + 3 + 4 + 5 = 15. Complete los espacios para que esto funcione:

def sum\_positive\_numbers(n):

    # The base case is n being smaller than 1

    if n < 1:

        return \_\_\_

    # The recursive case is adding this number to

    # the sum of the numbers smaller than this one.

    return \_\_\_ + sum\_positive\_numbers(\_\_\_)

print(sum\_positive\_numbers(3)) # Should be 6

print(sum\_positive\_numbers(5)) # Should be 15

def sum\_positive\_numbers(n):

    # The base case is n being smaller than 1

    if n < 1:

        return 0

    # The recursive case is adding this number to

    # the sum of the numbers smaller than this one.

    return n + sum\_positive\_numbers(n - 1)

print(sum\_positive\_numbers(3)) # Should be 6

print(sum\_positive\_numbers(5)) # Should be 15

#### Prueba de práctica: recursividad

Pregunta 1

Complete los espacios en blanco para que la función is\_power\_of devuelva si el número es una potencia de la base dada. Nota: se supone que la base es un número positivo. Consejo: para las funciones que devuelven un valor booleano, puede devolver el resultado de una comparación.

def is\_power\_of(number, base):

  # Base case: when number is smaller than base.

  if number < base:

    # If number is equal to 1, it's a power (base\*\*0).

    return \_\_

  # Recursive case: keep dividing number by base.

  return is\_power\_of(\_\_, \_\_\_)

print(is\_power\_of(8,2)) # Should be True

print(is\_power\_of(64,4)) # Should be True

print(is\_power\_of(70,10)) # Should be False

def is\_power\_of(number, base):

  # Base case: when number is smaller than base.

  if number < base:

    # If number is equal to 1, it's a power (base\*\*0).

    if number == 1:

      return True

    return False

  number /= base

  # Recursive case: keep dividing number by base.

  return is\_power\_of(number, base)

print(is\_power\_of(8,2)) # Should be True

print(is\_power\_of(64,4)) # Should be True

print(is\_power\_of(70,10)) # Should be False

Pregunta 2

La función count\_users cuenta recursivamente la cantidad de usuarios que pertenecen a un grupo en el sistema de la empresa, pasando por cada uno de los miembros de un grupo y si uno de ellos es un grupo, llamando recursivamente a la función y contando los miembros. ¡Pero tiene un error! ¿Puedes detectar el problema y solucionarlo?

def count\_users(group):

  count = 0

  for member in get\_members(group):

    count += 1

    if is\_group(member):

      count += count\_users(member)

  return count

print(count\_users("sales")) # Should be 3

print(count\_users("engineering")) # Should be 8

print(count\_users("everyone")) # Should be 18

def count\_users(group):

  count = 0

  for member in get\_members(group):

    count += 1

    if is\_group(member):

      count -= 1

      count += count\_users(member)

  return count

print(count\_users("sales")) # Should be 3

print(count\_users("engineering")) # Should be 8

print(count\_users("everyone")) # Should be 18

Pregunta 3

Implementar la función sum\_positive\_numbers, como una función recursiva que devuelve la suma de todos los números positivos entre el número n recibido y 1. Por ejemplo, cuando n es 3 debería devolver 1 + 2 + 3 = 6, y cuando n es 5 debería devuelve 1 + 2 + 3 + 4 + 5 = 15.

def sum\_positive\_numbers(n):

  return 0

print(sum\_positive\_numbers(3)) # Should be 6

print(sum\_positive\_numbers(5)) # Should be 15

def sum\_positive\_numbers(n):

  if n < 1:

    return 0

  return n + sum\_positive\_numbers(n-1)

print(sum\_positive\_numbers(3)) # Should be 6

print(sum\_positive\_numbers(5)) # Should be 15

#### Evaluación calificada del módulo 3

Pregunta 1

Complete los espacios en blanco de este código para imprimir los números del 1 al 7.

number = 1

while number \_\_\_ 7:

    print(number, end=" ")

    \_\_\_

number = 1

while number <= 7:

    print(number, end=" ")

    number += 1

Pregunta 2

La función show\_letters debería imprimir cada letra de una palabra en una línea separada. Complete los espacios en blanco para que eso suceda.

def show\_letters(word):

    for \_\_:

        print(\_\_)

show\_letters("Hello")

# Should print one line per letter

def show\_letters(word):

    for letter in word:

        print(letter)

show\_letters("Hello")

# Should print one line per letter

Pregunta 3

Complete la función dígitos (n) que devuelve cuántos dígitos tiene el número. Por ejemplo: 25 tiene 2 dígitos y 144 tiene 3 dígitos. **Consejo:** puede calcular los dígitos de un número dividiéndolo por 10 una vez por dígito hasta que no queden dígitos.

def digits(n):

    count = 0

    if n == 0:

      \_\_\_

    while (\_\_\_):

        count += 1

        \_\_\_

    return count

print(digits(25))   # Should print 2

print(digits(144))  # Should print 3

print(digits(1000)) # Should print 4

print(digits(0))    # Should print 1

def digits(n):

    count = 0

    if n == 0:

      return 1

    while (n >= 1):

        count += 1

        n /= 10

    return count

print(digits(25))   # Should print 2

print(digits(144))  # Should print 3

print(digits(1000)) # Should print 4

print(digits(0))    # Should print 1

Pregunta 4

Esta función imprime una tabla de multiplicar (donde cada número es el resultado de multiplicar el primer número de su fila por el número en la parte superior de su columna). Complete los espacios en blanco para que se imprima al llamar a multiplication\_table (1, 3):

1 2 3

2 4 6

3 6 9

def multiplication\_table(start, stop):

    for x in \_\_\_:

        for y in \_\_\_:

            print(str(x\*y), end=" ")

        print()

multiplication\_table(1, 3)

# Should print the multiplication table shown above

def multiplication\_table(start, stop):

    for x in range(start,stop+1):

        for y in range(start, stop+1):

            print(str(x\*y), end=" ")

        print()

multiplication\_table(1, 3)

# Should print the multiplication table shown above

Pregunta 5

La función de contador cuenta hacia atrás desde el inicio hasta el final cuando el inicio es mayor que el final, y cuenta desde el inicio hasta el final de lo contrario. Complete los espacios en blanco para que esto funcione correctamente.

def counter(start, stop):

    x = start

    if \_\_\_:

        return\_string = "Counting down: "

        while x >= stop:

            return\_string += str(x)

            if \_\_\_:

                return\_string += ","

            \_\_\_

    else:

        return\_string = "Counting up: "

        while x <= stop:

            return\_string += str(x)

            if \_\_\_:

                return\_string += ","

            \_\_\_

    return return\_string

print(counter(1, 10)) # Should be "Counting up: 1,2,3,4,5,6,7,8,9,10"

print(counter(2, 1)) # Should be "Counting down: 2,1"

print(counter(5, 5)) # Should be "Counting up: 5"

def counter(start, stop):

    x = start

    if start > stop:

        return\_string = "Counting down: "

        while x >= stop:

            return\_string += str(x)

            if \_\_\_:

                return\_string += ","

            x -= 1

    else:

        return\_string = "Counting up: "

        while x <= stop:

            return\_string += str(x)

            if len(return\_string) < stop:

                return\_string += ","

            x += 1

    return return\_string

print(counter(1, 10)) # Should be "Counting up: 1,2,3,4,5,6,7,8,9,10"

print(counter(2, 1)) # Should be "Counting down: 2,1"

print(counter(5, 5)) # Should be "Counting up: 5"

Pregunta 6

La función even\_numbers devuelve una cadena separada por espacios de todos los números positivos que son divisibles por 2, hasta el máximo que se pasa a la función inclusive. Por ejemplo, even\_numbers (6) devuelve "2 4 6". Complete el espacio en blanco para que esto funcione.

def even\_numbers(maximum):

    return\_string = ""

    for x in \_\_\_:

        return\_string += str(x) + " "

    return return\_string.strip()

print(even\_numbers(6))  # Should be 2 4 6

print(even\_numbers(10)) # Should be 2 4 6 8 10

print(even\_numbers(1))  # No numbers displayed

print(even\_numbers(3))  # Should be 2

print(even\_numbers(0))  # No numbers displayed

def even\_numbers(maximum):

    return\_string = ""

    for x in range(1, maximum+1):

        if (x % 2) == 0:

            return\_string += str(x) + " "

    return return\_string.strip()

print(even\_numbers(6))  # Should be 2 4 6

print(even\_numbers(10)) # Should be 2 4 6 8 10

print(even\_numbers(1))  # No numbers displayed

print(even\_numbers(3))  # Should be 2

print(even\_numbers(0))  # No numbers displayed

### Strings

#### Ejercicio (¿Qué es un string?)

Modifique la función double\_word para que devuelva la misma palabra repetida dos veces, seguida de la longitud de la nueva palabra duplicada. Por ejemplo, double\_word ("hello") debería devolver hellohello10.

def double\_word(word):

    return

print(double\_word("hello")) # Should return hellohello10

print(double\_word("abc"))   # Should return abcabc6

print(double\_word(""))      # Should return 0

def double\_word(word):

    return word \* 2 + str(len(word \* 2))

print(double\_word("hello")) # Should return hellohello10

print(double\_word("abc"))   # Should return abcabc6

print(double\_word(""))      # Should return 0

#### Ejercico (Las partes de un string)

¿Quieres intentarlo tú mismo? ¡Sé mi invitado! Modifique la función first\_and\_last para que devuelva True si la primera letra de la cadena es la misma que la última letra de la cadena, False si son diferentes. Recuerde que puede acceder a los caracteres mediante el mensaje [0] o el mensaje [-1]. Tenga cuidado con cómo maneja la cadena vacía, que debería devolver True ya que nada es igual a nada.

def first\_and\_last(message):

    return False

print(first\_and\_last("else"))

print(first\_and\_last("tree"))

print(first\_and\_last(""))

def first\_and\_last(message):

    if message == "":

        return True

    elif message[0] == message[-1]:

        return True

    return False

print(first\_and\_last("else"))

print(first\_and\_last("tree"))

print(first\_and\_last(""))

#### Ejercicio (Crear nuevas cadenas)

¡Intente usar el método de índice usted mismo ahora!

Usando el método de índice, averigüe la posición de "x" en "supercalifragilisticexpialidocious".

word = "supercalifragilisticexpialidocious"

print(\_\_\_)

word = "supercalifragilisticexpialidocious"

print(word.index("x"))

#### Ejercicio (Más métodos de cadena)

¿Quieres probar algunos métodos de cuerdas tú mismo? ¡Darle una oportunidad!

Rellena los huecos de la función de iniciales para que devuelva las iniciales de las palabras contenidas en la frase recibida, en mayúsculas. Por ejemplo: "Universal Serial Bus" debe devolver "USB"; "red de área local" debe devolver "LAN".

def initials(phrase):

    words = phrase.\_\_\_

    result = ""

    for word in words:

        result += \_\_\_

    return \_\_\_

print(initials("Universal Serial Bus")) # Should be: USB

print(initials("local area network")) # Should be: LAN

print(initials("Operating system")) # Should be: OS

def initials(phrase):

    words = phrase.split()

    result = ""

    for word in words:

        result += word[0].upper()

    return result

print(initials("Universal Serial Bus")) # Should be: USB

print(initials("local area network")) # Should be: LAN

print(initials("Operating system")) # Should be: OS

#### Ejercicio (Formateo de cadenas)

Modifique la función student\_grade utilizando el método de formato, de modo que devuelva la frase "X recibió Y% en el examen". Por ejemplo, student\_grade ("Reed", 80) debería devolver "Reed recibió el 80% en el examen".

def student\_grade(name, grade):

    return ""

print(student\_grade("Reed", 80))

print(student\_grade("Paige", 92))

print(student\_grade("Jesse", 85))

def student\_grade(name, grade):

    return "{} received {}% on the exam".format(name, grade)

print(student\_grade("Reed", 80))

print(student\_grade("Paige", 92))

print(student\_grade("Jesse", 85))

#### Prueba de práctica: Strings

Pregunta 1

La función is\_palindrome comprueba si una cadena es un palíndromo. Un palíndromo es una cadena que se puede leer por igual de izquierda a derecha o de derecha a izquierda, omitiendo espacios en blanco e ignorando las mayúsculas. Ejemplos de palíndromos son palabras como kayak y radar, y frases como "Nunca pares o impares". Complete los espacios en blanco en esta función para devolver True si la cadena pasada es un palíndromo, False si no.

def is\_palindrome(input\_string):

    # We'll create two strings, to compare them

    new\_string = ""

    reverse\_string = ""

    # Traverse through each letter of the input string

    for \_\_\_:

        # Add any non-blank letters to the

        # end of one string, and to the front

        # of the other string.

        if \_\_\_:

            new\_string = \_\_\_

            reverse\_string = \_\_\_

    # Compare the strings

    if \_\_\_:

        return True

    return False

print(is\_palindrome("Never Odd or Even")) # Should be True

print(is\_palindrome("abc")) # Should be False

print(is\_palindrome("kayak")) # Should be True

def is\_palindrome(input\_string):

    # We'll create two strings, to compare them

    new\_string = ""

    reverse\_string = ""

    # Traverse through each letter of the input string

    for letter in input\_string.lower():

        # Add any non-blank letters to the

        # end of one string, and to the front

        # of the other string.

        if letter != " ":

            new\_string = new\_string + letter

            reverse\_string = letter + reverse\_string

    # Compare the strings

    if new\_string == reverse\_string:

        return True

    return False

print(is\_palindrome("Never Odd or Even")) # Should be True

print(is\_palindrome("abc")) # Should be False

print(is\_palindrome("kayak")) # Should be True

Pregunta 2

Usando el método de formato, complete los espacios en la función convert\_distance para que devuelva la frase "X millas es igual a Y km", con Y teniendo solo 1 lugar decimal. Por ejemplo, convert\_distance (12) debería devolver "12 millas equivalen a 19,2 km".

def convert\_distance(miles):

    km = miles \* 1.6

    result = "{} miles equals {\_\_\_} km".\_\_\_

    return result

print(convert\_distance(12)) # Should be: 12 miles equals 19.2 km

print(convert\_distance(5.5)) # Should be: 5.5 miles equals 8.8 km

print(convert\_distance(11)) # Should be: 11 miles equals 17.6 km

def convert\_distance(miles):

    km = miles \* 1.6

    result = "{} miles equals {:.1f} km".format(miles, km)

    return result

print(convert\_distance(12)) # Should be: 12 miles equals 19.2 km

print(convert\_distance(5.5)) # Should be: 5.5 miles equals 8.8 km

print(convert\_distance(11)) # Should be: 11 miles equals 17.6 km

Pregunta 4

Complete los espacios en la función de etiqueta de nombre para que use el método de formato para devolver first\_name y la primera inicial de last\_name seguida de un punto. Por ejemplo, la etiqueta de nombre ("Jane", "Smith") debe devolver "Jane S."

def nametag(first\_name, last\_name):

    return("\_\_\_.".format(\_\_\_))

print(nametag("Jane", "Smith"))

# Should display "Jane S."

print(nametag("Francesco", "Rinaldi"))

# Should display "Francesco R."

print(nametag("Jean-Luc", "Grand-Pierre"))

# Should display "Jean-Luc G."

def nametag(first\_name, last\_name):

    return("{} {}.".format(first\_name, last\_name[0]))

print(nametag("Jane", "Smith"))

# Should display "Jane S."

print(nametag("Francesco", "Rinaldi"))

# Should display "Francesco R."

print(nametag("Jean-Luc", "Grand-Pierre"))

# Should display "Jean-Luc G."

Pregunta 5

La función replace\_ending reemplaza la cadena anterior en una oración con la cadena nueva, pero solo si la oración termina con la cadena anterior. Si hay más de una aparición de la cadena anterior en la oración, solo se reemplaza la que está al final, no todas. Por ejemplo, replace\_ending ("abcabc", "abc", "xyz") debería devolver abcxyz, no xyzxyz o xyzabc. La comparación de cadenas distingue entre mayúsculas y minúsculas, por lo que replace\_ending ("abcabc", "ABC", "xyz") debería devolver abcabc (no se realizaron cambios).

def replace\_ending(sentence, old, new):

    # Check if the old string is at the end of the sentence

    if \_\_\_:

        # Using i as the slicing index, combine the part

        # of the sentence up to the matched string at the

        # end with the new string

        i = \_\_\_

        new\_sentence = \_\_\_

        return new\_sentence

    # Return the original sentence if there is no match

    return sentence

print(replace\_ending("It's raining cats and cats", "cats", "dogs"))

# Should display "It's raining cats and dogs"

print(replace\_ending("She sells seashells by the seashore", "seashells", "donuts"))

# Should display "She sells seashells by the seashore"

print(replace\_ending("The weather is nice in May", "may", "april"))

# Should display "The weather is nice in May"

print(replace\_ending("The weather is nice in May", "May", "April"))

# Should display "The weather is nice in April"

def replace\_ending(sentence, old, new):

    # Check if the old string is at the end of the sentence

    if sentence.endswith(old):

        # Using i as the slicing index, combine the part

        # of the sentence up to the matched string at the

        # end with the new string

        i = len(old)

        new\_sentence = sentence[0:-i] + new

        return new\_sentence

    # Return the original sentence if there is no match

    return sentence

print(replace\_ending("It's raining cats and cats", "cats", "dogs"))

# Should display "It's raining cats and dogs"

print(replace\_ending("She sells seashells by the seashore", "seashells", "donuts"))

# Should display "She sells seashells by the seashore"

print(replace\_ending("The weather is nice in May", "may", "april"))

# Should display "The weather is nice in May"

print(replace\_ending("The weather is nice in May", "May", "April"))

# Should display "The weather is nice in April"