TIC

Ciclos

Ahora imagine que quiere saber si en un listado de 100 años hay años bisiestos, tendría que repetir el proceso realizado en la ejercitación anterior para cada uno de los elementos de la lista. Ahora imagine que quiere saber todos los años bisiestos hasta 2024.

Es un proceso repetitivo que no es eficiente hacerlo manualmente uno por uno, se requiere de una estructura que permita repetir un grupo de instrucciones según la necesidad del usuario.
Para realizar tareas de este estilo, Python implementa dos herramientas, el ciclo for y while

Ciclo while

La sentencia while se usa para la ejecución repetida siempre que una expresión sea verdadera. Cuenta con la senntencia break que permite finalizar la iteración en el momento que se determine, por ejemplo en un ciclo de contar hasta 100 parar cuando llegue a un número determinado.

También cuenta con continue para solo saltar la iteración actual y pasar a la siguiente.

Finalmente cuenta con la sentencia else que es opcional y se ejecuta cuando la expresión que evalúa el while no es más verdadera.

```
mientras haya algo que hacer
    hazlo
```

Toma en cuenta que este registro también declara que, si no hay nada que hacer, nada ocurrirá.

```
while conditional_expression:
   instruction
   ...
```

La sentencia es muy similar al if, solo que tiene la posibilidad de repetirse un número determinado de veces (o incluso indefinidamente) Ahora, es importante recordar que:

- si deseas ejecutar más de una sentencia dentro de un while, debes (como con if) poner sangría a todas las instrucciones de la misma manera.
- una instrucción o conjunto de instrucciones ejecutadas dentro del while se llama el cuerpo del bucle.
- si la condición es False (igual a cero) tan pronto como se compruebe por primera vez, el cuerpo no se ejecuta ni una sola vez (ten en cuenta la analogía de no tener que hacer nada si no hay nada que hacer).
- el cuerpo debe poder cambiar el valor de la condición, porque si la condición es True al principio, el cuerpo podría funcionar continuamente hasta el infinito (bucle infinito). Observa que hacer una cosa generalmente disminuye la cantidad de cosas por hacer.

Ejecuta esto dentro de un computador y observe el resultado, debe recordar detener la ejecución con las teclas CTRL + C. Mucho cuidado, puede quedarse sin memoria.

```
while True:
    print("Estoy atrapado dentro de un bucle.")
```

Aquí entra algo muy importante y es la condición de parada, el cómo mi código puede salir del bucle o determinar que las
iteraciones han terminado.

A continuación se presenta un código que determina el número más grande de una secuencia de números ingresados por el usuario hasta que se cumpla una condición de salida. Mientras la condición no se cumpla el ciclo se seguirá ejecutando.

```
[ ]  # Almacena el actual número más grande aquí.
largest_number = -999999999

# Ingresa el primer valor.
number = int(input("Introduce un número o escribe -1 para detener: "))

# Si el número no es igual a -1, continuaremos
while number != -1:
    # ¿Es el número más grande que el valor de largest_number?
    if number > largest_number:
        # Sí si, se actualiza largest_number.
        largest_number = number
    # Ingresa el siguiente número.
    number = int(input("Introduce un número o escribe -1 para detener: "))

# Imprime el número más grande.
print("El número más grande es:", largest_number)
```

O este otro código que saluda mientras el usuario quiera ser saludado

```
[ ]  want_greet = 'S'

     valid_options = 0

     while want_greet == 'S':

         print('Hola qué tal!')

         want_greet = input('¿Quiere otro saludo? [S/N] ')

         if want_greet not in 'SN':

             print('No le he entendido pero le saludo')

             want_greet = 'S'

             continue

         valid_options += 1

     print(f'{valid_options} respuestas válidas')

     print('Que tenga un buen día')
```

```
Hola qué tal!
¿Quiere otro saludo? [S/N] a
No le he entendido pero le saludo
Hola qué tal!
¿Quiere otro saludo? [S/N] S
Hola qué tal!
¿Quiere otro saludo? [S/N] S
Hola qué tal!
¿Quiere otro saludo? [S/N] N
3 respuestas válidas
Que tenga un buen día
```

Ejercitación

Implementar un código que imprima los números factoriales, utilizando while, hasta un número determinado.
Imprimir la sucesión de Fibonacci hasta el número 20. Cómo implementaría la condición de parada?
Implemente un código con dos contadores, uno para números pares y otro para números impares, ahora haga que el usuario ingrese tantos números como desee y sume a cada contador cuando corresponda, recuerde definir la condición de parada, por ejemplo, cuando se ingrese 0.

Ejercitación

Un mago junior ha elegido un número secreto. Lo ha escondido en una variable llamada secret_number. Quiere que todos los que ejecutan su programa jueguen el juego divina el número secreto, y adivina qué número ha elegido para ellos. ¡Quiénes no adivinen el número quedarán atrapados en un bucle sin fin para siempre! Desafortunadamente, él no sabe cómo completar el código.

Tu tarea es ayudar al mago a completar el código en el editor de tal manera que el código:

pedirá al usuario que ingrese un número entero; utilizará un bucle while; comprobará si el número ingresado por el usuario es el mismo que el número escogido por el mago. Si el número elegido por el usuario es diferente al número secreto del mago, el usuario debería ver el mensaje "¡Ja, ja! ¡Estás atrapado en mi bucle!" y se le solicitará que ingrese un número nuevamente. Si el número ingresado por el usuario coincide con el número escogido por el mago, el número debe imprimirse en la pantalla, y el mago debe decir las siguientes palabras: "¡Bien hecho, muggle! Eres libre ahora."

```
[ ]  secret_number = 777

     print(
     """
     +================================+
     | ¡Bienvenido a mi juego, muggle!|
     | Introduce un número entero     |
     | y adivina qué número he        |
     | elegido para ti.               |
     |¿Cuál es el número secreto?     |
     +================================+
     """)

     #Ingresa el código aquí
```

Ciclo For

El ciclo For es otra de las formas con las que cuenta Python para realizar iteraciones sobre secuencias o iterables como lo son listas, tuplas, diccionarios, sets e incluso cadenas.
Funciona menos como ciclos for en otros lenguajes de programación y más como un método iterador en lenguajes orientados a objetos.
Dentro del bloque for es posible realizar diferentes operaciones aplicadas a cada uno de los elementos del iterable y combinar todos los elementos vistos hasta el momento.
Una diferencia entre el for y el while es que el primero cuenta la iteraciones hasta completar la condición dada mientras que el segundo se ejecuta mientras la condición dada se cumpla.
Y una de sus características es que podemos recorrer estructuras de datos más complejas, dato por dato.

```
for i in range(100):
    # do_something()
    pass
```

Existen algunos elementos nuevos. Déjanos contarte sobre ellos:

- La palabra reservada for abre el bucle for; nota – No hay condición después de eso; no tienes que pensar en las condiciones, ya que se verifican internamente, sin ninguna intervención.
- cualquier variable después de la palabra reservada for es la variable de control del bucle; cuenta los giros del bucle y lo hace automáticamente.
- la palabra reservada in introduce un elemento de sintaxis que describe el rango de valores posibles que se asignan a la variable de control.
- la función range() (esta es una función muy especial) es responsable de generar todos los valores deseados de la variable de control; en nuestro ejemplo, la función creará (incluso podemos decir que alimentará el bucle con) valores subsiguientes del siguiente conjunto: 0, 1, 2 .. 97, 98, 99; nota: en este caso, la función range() comienza su trabajo desde 0 y lo finaliza un paso (un número entero) antes del valor de su argumento.

Nota: la palabra clave pass dentro del cuerpo del bucle – no hace nada en absoluto; es una instrucción vacía – la olocamos aquí porque la sintaxis del bucle for exige al menos una instrucción dentro del cuerpo.

Ejercitación

Qué se obtiene de ejecutar la siguiente instrucción?

```
for i in range(10):
    print("El valor de i es", i)
```

El ciclo for cuenta con una sentencia break que detiene la iteración al detectar su llamado.

Tambien con continueque finaliza la iteración actual pero no rompe el ciclo.

```
[ ]  #Iterando la lista de estudiantes

     listado_estudiantes = ["Alice", "Joha", "Damien", "Javi"]

     for estudiante in listado_estudiantes:
       print(estudiante);

     nombre_asignatura = "Matemáticas"

     print('\n')

     for x in nombre_asignatura:
       if(x.lower() == 'c'):
         break
       print('\n', x)

     print('\n')

     for x in nombre_asignatura:
       if(x.lower() == 't'):
         continue
       print(x)
```

Ahora, el ciclo for cuenta con la sentencia range() donde es posible definir el inicio y el final de la iteración, sin necesidad de estar recorriendo un iterable.

La función range()cuenta con hasta 3 parámetros, cuando solo se utiliza solo uno, este define el limite superior, es decir el final del ciclo, sin incluir, el inicio es por defecto 0. Cuando se usan dos parámetros, el primero define el límite inferior o el inicio del ciclo y el segundo el final del ciclo, igualmente sin incluirlo y, finalmente el tercer parámetro indica el incremento, por defecto el incremento es de 1, pero el valor puede ser modificado.

Nota: la función range() solo acepta enteros como argumentos y genera secuencias de enteros.

```
range(2, 9, 2) #Inicia en 2 avanzando de 2 en 2 hasta llegar a 8, ya que el 9 no lo incluye
```

El ciclo for cuenta con una sentencia else que se ejecuta al finalizar la última iteración

También cuenta con la posibilidad de romper el ciclo utilizando break, teniendo en cuenta que el else no se ejecuta cuando esto ocurre.

Se cuenta con la posibilidad de hacer ciclos for anidados y de utilizar la sentencia pass para avanzar sin definir acciones dentro del bloque.

```
[ ]  for x in range(2, 9, 2):
       print(x)
     else:
       print('Solo pares')


     listado_estudiantes = ["Alice", "Joha", "Damien", "Javi"]
     notas_estudiantes = [4, 4.5, 5, 3.8]

     for estudiante in listado_estudiantes:
       for nota in notas_estudiantes:
         print(estudiante, nota)
       else:
         print("algo no quedó bien")
     else:
       print('Hay que repetir todo')
```

Ejercitación contando Mississippis

¿Sabes lo que es Mississippi? Bueno, es el nombre de uno de los estados y ríos en los Estados Unidos. El río Mississippi tiene aproximadamente 2,340 millas de largo, lo que lo convierte en el segundo río más largo de los Estados Unidos (el más largo es el río Missouri). ¡Es tan largo que una sola gota de agua necesita 90 días para recorrer toda su longitud!

La palabra Mississippi también se usa para un propósito ligeramente diferente: para contar mississippily (mississippimente).

Si no estás familiarizado con la frase, estamos aquí para explicarte lo que significa: se utiliza para contar segundos.

La idea detrás de esto es que agregar la palabra Mississippi a un número al contar los segundos en voz alta hace que suene más cercano al reloj, y por lo tanto "un Mississippi, dos Mississippi, tres Mississippi" tomará aproximadamente unos tres segundos reales de tiempo. A menudo lo usan los niños que juegan al escondite para asegurarse de que el buscador haga un conteo honesto.

Tu tarea es muy simple aquí: escribe un programa que use un bucle for para "contar de forma mississippi" hasta cinco.
Habiendo contado hasta cinco, el programa debería imprimir en la pantalla el mensaje final "¡Listos o no, ahí voy!"

```
[ ]  import time

     # Escribe un bucle for que cuente hasta cinco.
         # Cuerpo del bucle: imprime el número de iteración del bucle y la palabra "Mississippi".
         # Cuerpo del bucle, emplea : time.sleep(1)

     # Escribe una función print con el mensaje final.
```

El paquete timey el método sleep() son utilizados para generar un retraso de 1 segundo en este caso, por ahora solo debe preocuparse por seguir las instrucciones indicadas en los comentarios.

TIC

Ejercitación Palabra secreta, palabra segura

Crear un programa que pida al usuario ingresar palabras, las palabras se almacenarán en un variable concatenando cada una con un espacio en blanco, el programa debe implementar una palabra que no se agregue cuando se diga que se conocerá como la palabra_secreta y otra palabra que haga que se finalice toda la ejecución y se rompa el ciclo llamada la palabra_segura.
Antes de romper todo el ciclo, debe imprimir todas las palabras ingresadas por el usuario.

Ejercitación Devorador de vocales

Ahora debe escribir un programa que dada una palabra, imprima línea por línea las letras sin las vocales

```
Entrada: Gregory


Salida:
G
R
G
R
Y
```

Ejercitación y el else?

Los ciclos for y while tienen una sentencia final llamada else, de igual funcionamiento al del bloque if-else, puede descubrir cómo aplicarla aquí?

Ejercitación DMG la pirámide

Qué tal uno más complejo

Debe crear un programa que determine qué altura pueda alcanzar una pirámide con un número X de bloques construida por un niño y su padre, teniendo en cuenta:

- cada capa inferior contiene un bloque más que la capa superior.
- si los constructores no tienen la cantidad suficiente de bloques y no pueden completar la siguiente capa, terminan su trabajo inmediatamente.

```
Entrada: 6
Salida: "La altura de la pirámide es: 3"

Entrada: 20
Salida: "La altura de la pirámide es: 5"

Entrada: 1000
Salida: "La altura de la pirámide es: 44"
```

Ejercitación Collatz

En 1937, un matemático alemán llamado Lothar Collatz formuló una hipótesis intrigante (aún no se ha comprobado) que se puede describir de la siguiente manera:

• toma cualquier número entero que no sea negativo y que no sea cero y asígnale el nombre c0; • si es par, evalúa un nuevo c0 como c0 ÷ 2; • de lo contrario, si es impar, evalúe un nuevo c0 como 3 × c0 + 1;
• si c0 ≠ 1, salta al punto 2.

La hipótesis dice que, independientemente del valor inicial de
c0, el valor siempre tiende a 1.

Por supuesto, es una tarea extremadamente compleja usar una computadora para probar la hipótesis de cualquier número natural (incluso puede requerir inteligencia artificial), pero puede usar Python para verificar algunos números individuales. Tal vez incluso encuentres el que refutaría la hipótesis.

Escribe un programa que lea un número natural y ejecute los pasos anteriores siempre que c0 sea diferente de 1. También queremos que cuente los pasos necesarios para lograr el objetivo. Tu código también debe mostrar todos los valores intermedios de c0.
Sugerencia: la parte más importante del problema es como transformar la idea de Collatz en un bucle while– esta es la clave del éxito.

```
Entrada: 15
Salida:
46
46
70
35
106
53
160
80
40
20
10
5
16
8
4
2
1
pasos = 17
```