DIRECCIÓN GENERAL DE UNIVERSIDADES TECNOLÓGICAS y POLITÉCNICAS

Universidad Tecnológica del Norte de Guanajuato
Organismo Público Descentralizado del Gobierno del Estado
"Educación y progreso para la vida"

PE1: Módulo 3: Valores booleanos, ejecución condicional, bucles, listas y su procesamiento, operadores lógicos y de bit a bit

Profesor:
Javier Jesús Torres Yañez

Unidad 1:
Principios de Programación de Redes

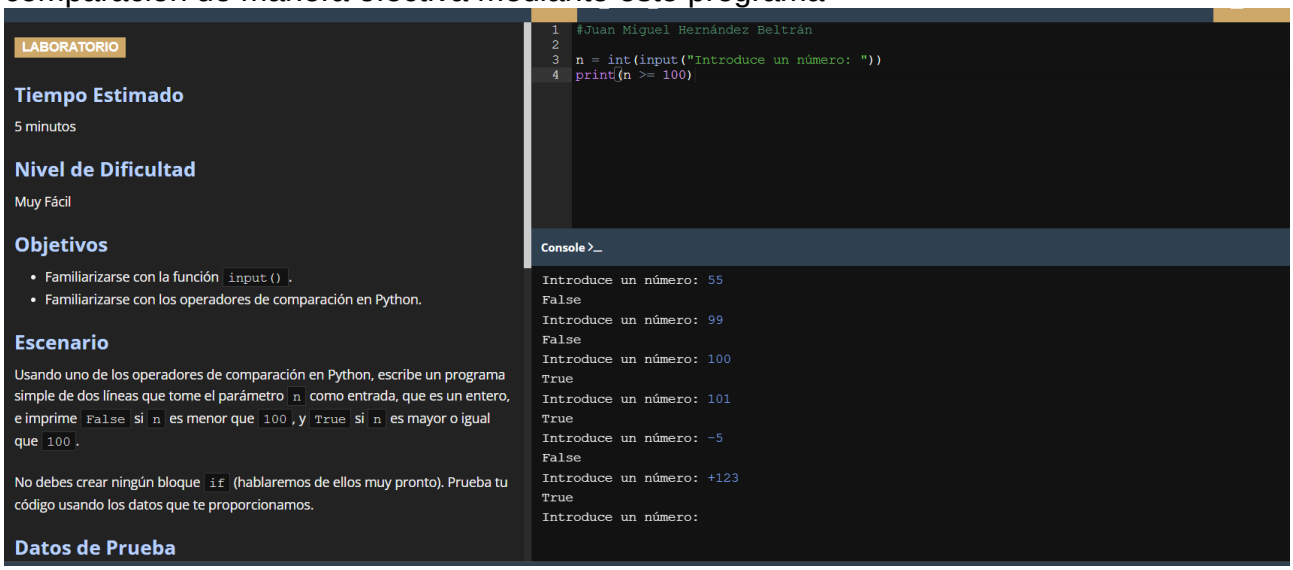
Alumno:
Juan Miguel Hernández Beltrán 1223100787

GRUPO:
GIR0642

TSU Infraestructura de Redes Digitales

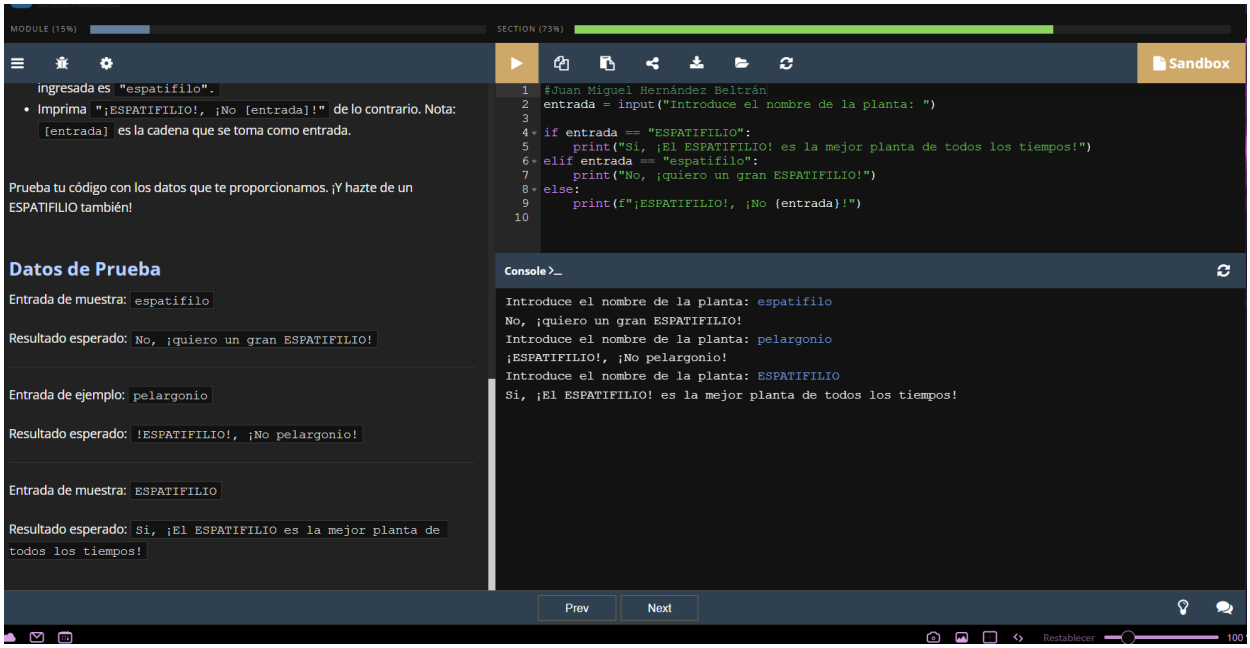
3.1.1.4 LABORATORIO: Preguntas y respuestas

En esta actividad, aprendí a utilizar la función input() y a aplicar operadores de comparación de manera efectiva mediante este programa



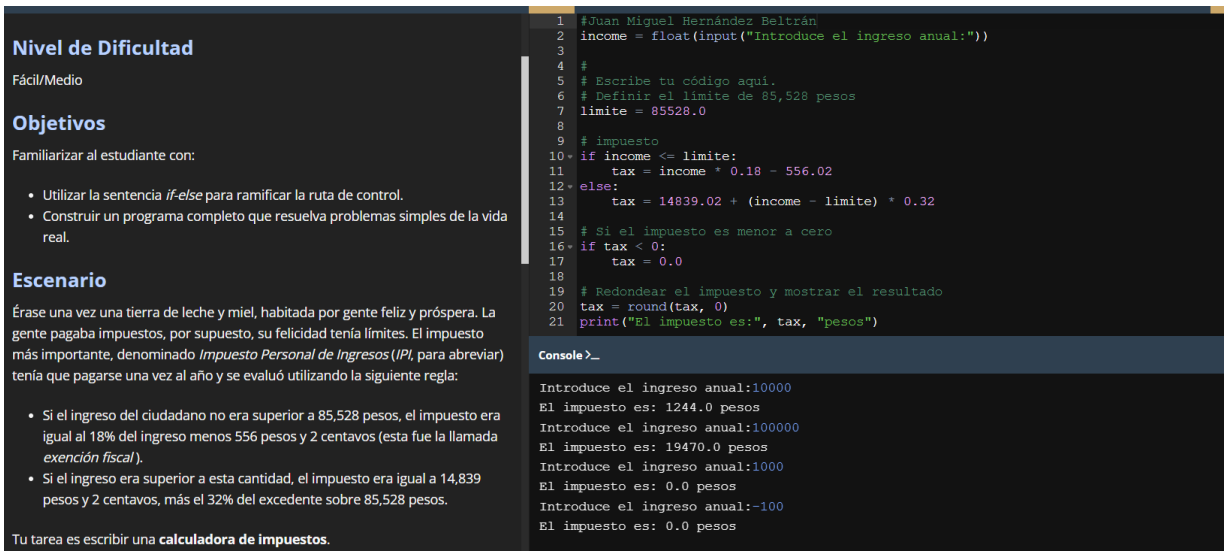
3.1.1.10 LABORATORIO: Operadores de comparación y ejecución condicional

En esta actividad aprendí a implementar la ejecución condicional utilizando la función input(). Al crear un programa que responde a diferentes entradas relacionadas con la planta Espatifilo.



3.1.1.11 LABORATORIO: Fundamentos de la sentencia if-else

En esta actividad, aprendí a utilizar la sentencia if-else para ramificar la lógica de un programa en Python.



Nivel de Dificultad

Fácil/Medio

Objetivos

Familiarizar al estudiante con:

- Utilizar la sentencia *if-else* para ramificar la ruta de control.
- Construir un programa completo que resuelva problemas simples de la vida real.

Escenario

Érase una vez una tierra de leche y miel, habitada por gente feliz y próspera. La gente pagaba impuestos, por supuesto, su felicidad tenía límites. El impuesto más importante, denominado *Impuesto Personal de Ingresos* (*IPI*, para abreviar) tenía que pagarse una vez al año y se evaluó utilizando la siguiente regla:

- Si el ingreso del ciudadano no era superior a 85,528 pesos, el impuesto era igual al 18% del ingreso menos 556 pesos y 2 centavos (esta fue la llamada *exención fiscal*).
- Si el ingreso era superior a esta cantidad, el impuesto era igual a 14,839 pesos y 2 centavos, más el 32% del excedente sobre 85,528 pesos.

Tu tarea es escribir una **calculadora de impuestos**.

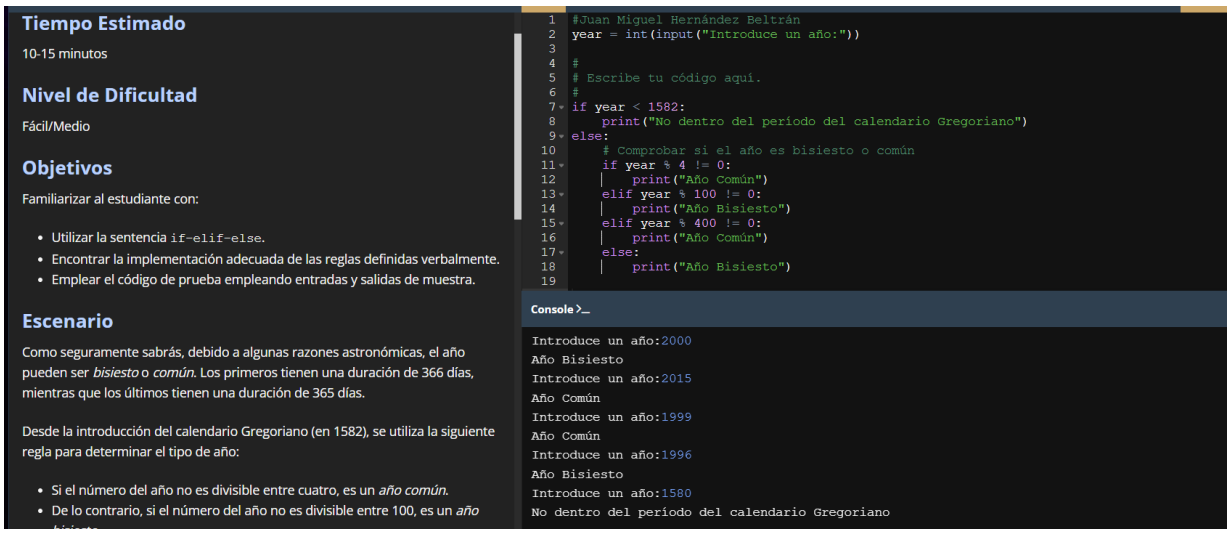
```python
#Juan Miguel Hernández Beltrán
income = float(input("Introduce el ingreso anual:"))

#
# Escribe tu código aquí.
# Definir el límite de 85,528 pesos
limite = 85528.0

# impuesto
if income <= limite:
    tax = income * 0.18 - 556.02
else:
    tax = 14839.02 + (income - limite) * 0.32

# Si el impuesto es menor a cero
if tax < 0:
    tax = 0.0

# Redondear el impuesto y mostrar el resultado
tax = round(tax, 0)
print("El impuesto es:", tax, "pesos")
```

Console >_

```
Introduce el ingreso anual:10000
El impuesto es: 1244.0 pesos
Introduce el ingreso anual:100000
El impuesto es: 19470.0 pesos
Introduce el ingreso anual:1000
El impuesto es: 0.0 pesos
Introduce el ingreso anual:-100
El impuesto es: 0.0 pesos
```

3.1.1.12 LABORATORIO: Fundamentos de la sentencia if-elif-else

En esta actividad, aprendí a utilizar la sentencia if-elif-else para implementar reglas lógicas al desarrollar un programa que determina si un año es bisiesto o común.



Tiempo Estimado

10-15 minutos

Nivel de Dificultad

Fácil/Medio

Objetivos

Familiarizar al estudiante con:

- Utilizar la sentencia `if-elif-else`.
- Encontrar la implementación adecuada de las reglas definidas verbalmente.
- Emplear el código de prueba empleando entradas y salidas de muestra.

Escenario

Como seguramente sabrás, debido a algunas razones astronómicas, el año pueden ser *bisiesto* o *común*. Los primeros tienen una duración de 366 días, mientras que los últimos tienen una duración de 365 días.

Desde la introducción del calendario Gregoriano (en 1582), se utiliza la siguiente regla para determinar el tipo de año:

- Si el número del año no es divisible entre cuatro, es un *año común*.
- De lo contrario, si el número del año no es divisible entre 100, es un *año*

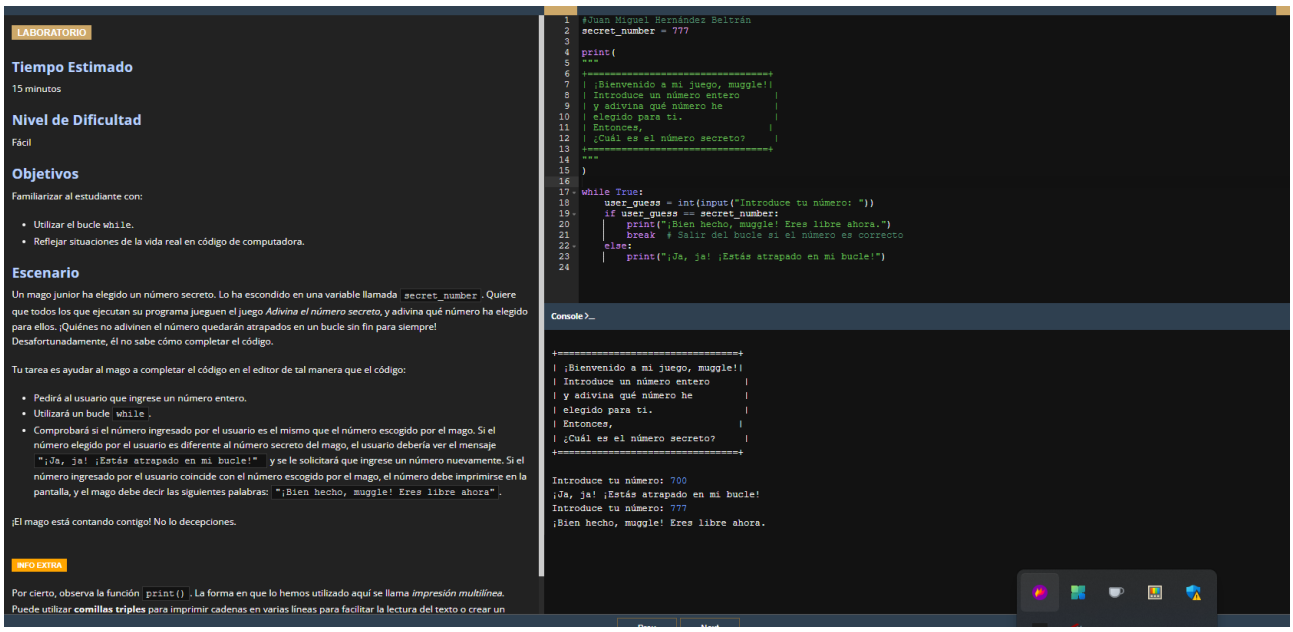
```python
#Juan Miguel Hernández Beltrán
year = int(input("Introduce un año:"))

#
# Escribe tu código aquí.
#
if year < 1582:
    print("No dentro del período del calendario Gregoriano")
else:
    # Comprobar si el año es bisiesto o común
    if year % 4 != 0:
        print("Año Común")
    elif year % 100 != 0:
        print("Año Bisiesto")
    elif year % 400 != 0:
        print("Año Común")
    else:
        print("Año Bisiesto")

```

Console >_

```
Introduce un año:2000
Año Bisiesto
Introduce un año:2015
Año Común
Introduce un año:1999
Año Común
Introduce un año:1996
Año Bisiesto
Introduce un año:1580
No dentro del período del calendario Gregoriano
```

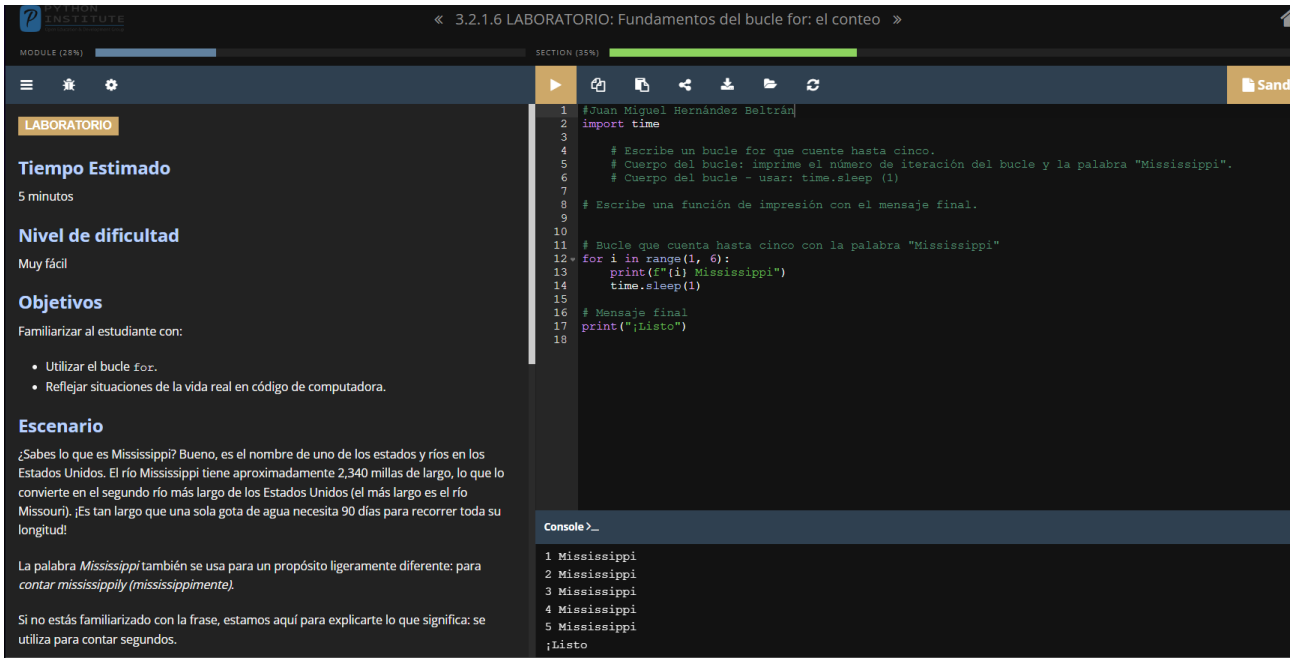
3.2.1.3 LABORATORIO: Lo esencial del bucle while - Adivina el número secreto

En esta actividad, aprendí a utilizar el bucle while para crear un juego interactivo en Python.



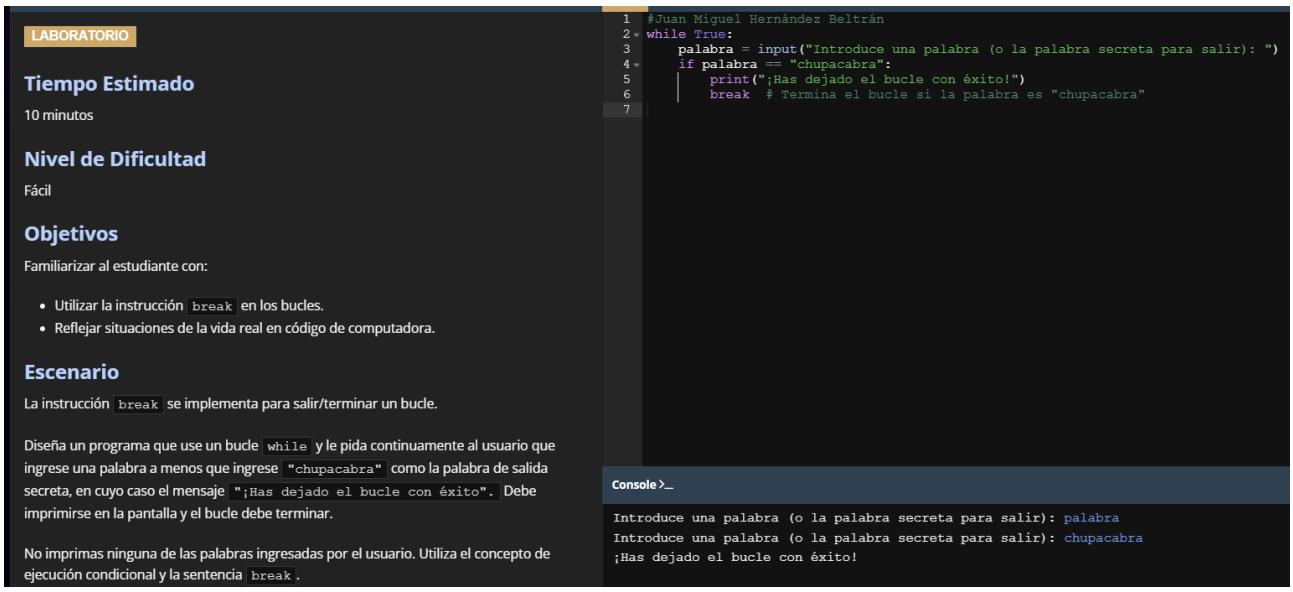
3.2.1.6 LABORATORIO: Fundamentos del bucle for: el conteo

En esta actividad, aprendí a utilizar un bucle for para implementar un conteo secuencial al crear un programa que cuenta



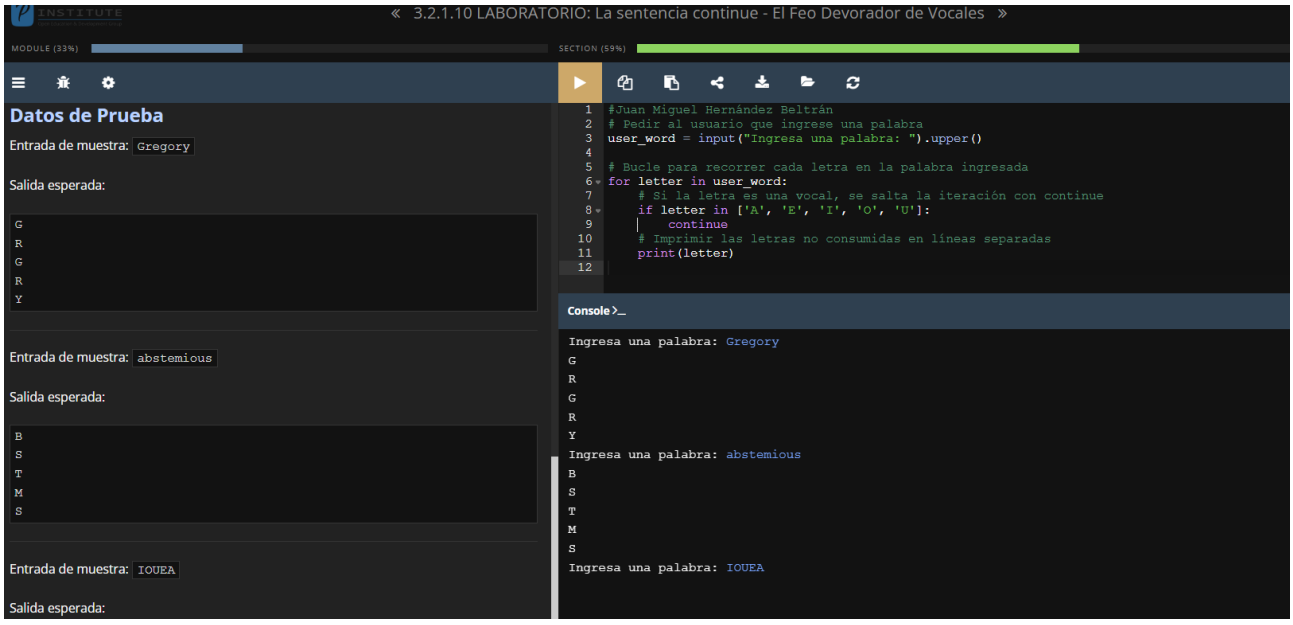
3.2.1.9 LABORATORIO: La sentencia break - Atascado en un bucle

En esta actividad, aprendí a utilizar la instrucción break en un bucle while para controlar el flujo de un programa.



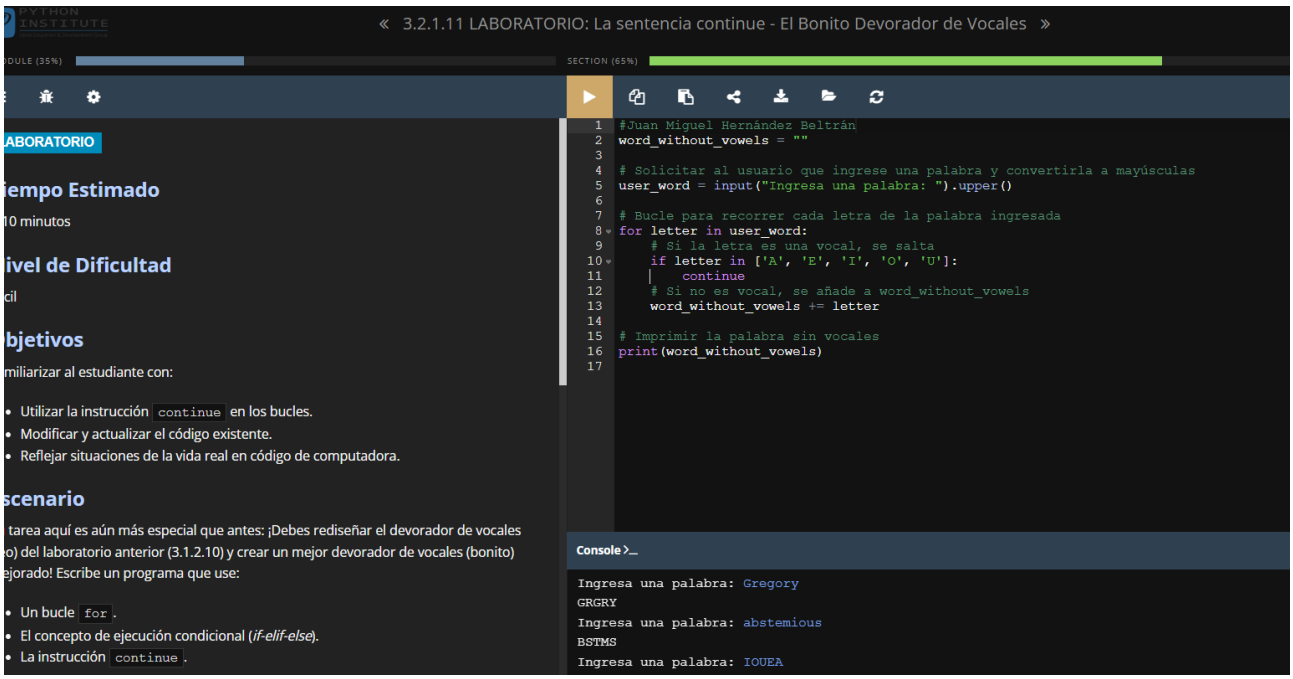
3.2.1.10 LABORATORIO: La sentencia continue - El Feo Devorador de Vocales

En esta actividad, aprendí a utilizar la instrucción continue dentro de un bucle for para omitir ciertas iteraciones al crear un "devorador de vocales" que implementé la lógica condicional para eliminar las vocales de una palabra ingresada por el usuario



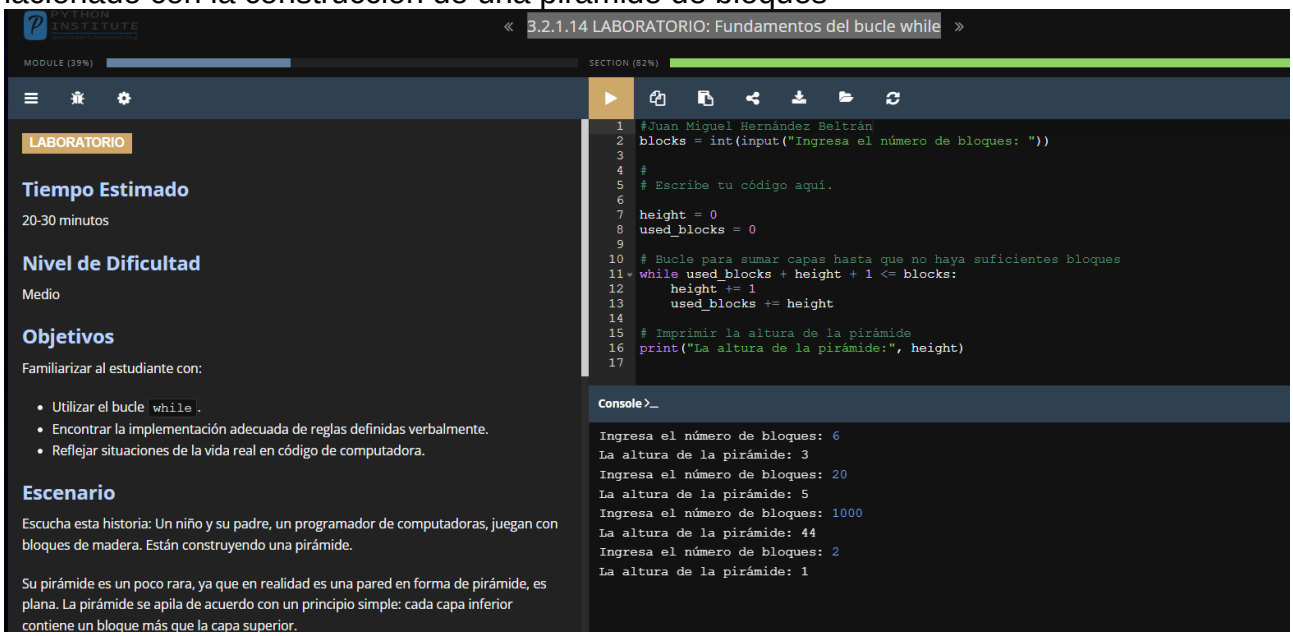
3.2.1.11 LABORATORIO: La sentencia continue - El Bonito Devorador de Vocales

En esta actividad, aprendí a rediseñar un programa para crear un "devorador de vocales" utilizando un bucle for y la instrucción continue, implementé la lógica condicional para eliminar las vocales de una palabra ingresada por el usuario.



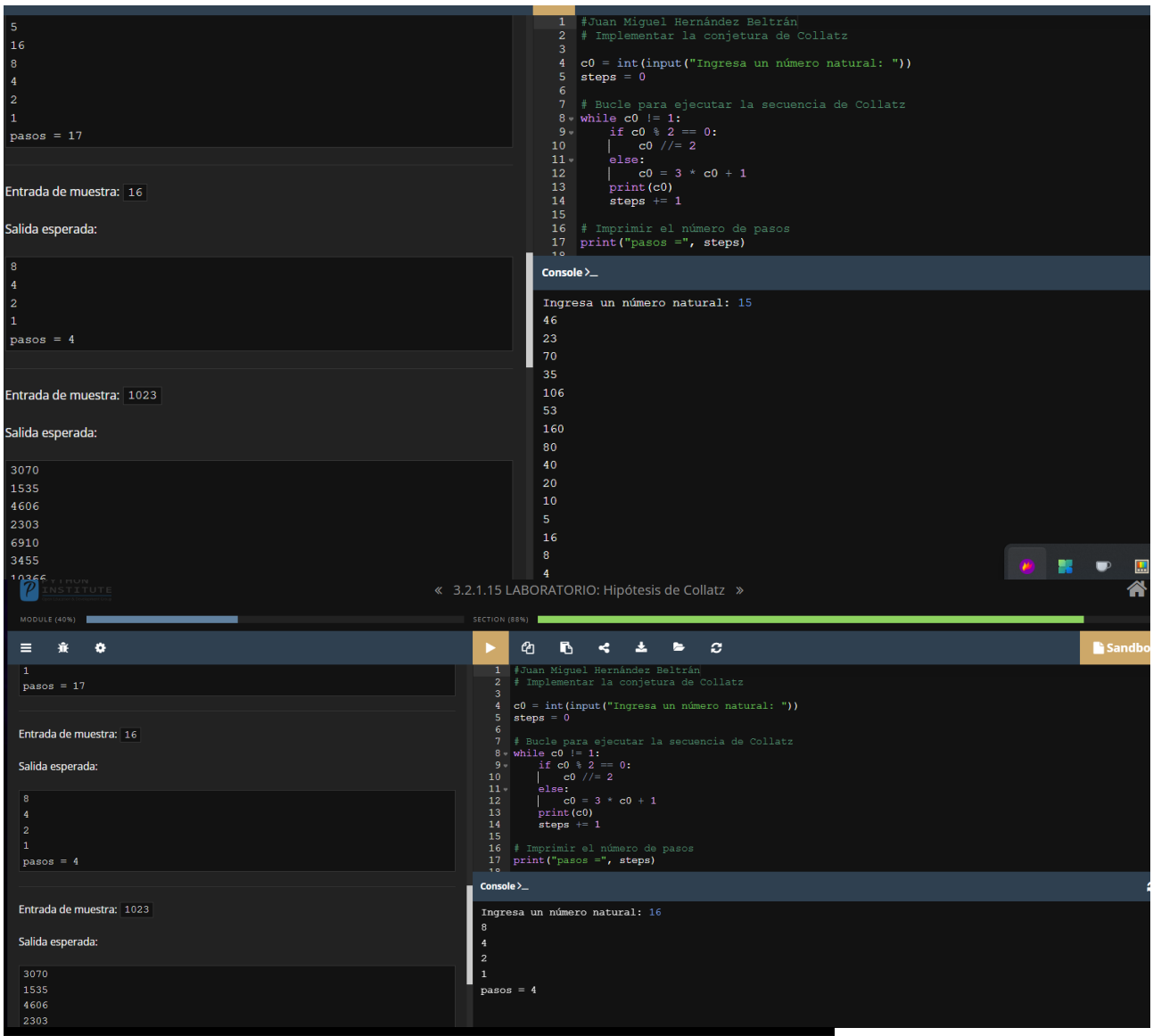
3.2.1.14 LABORATORIO: Fundamentos del bucle while

En esta actividad, aprendí a utilizar un bucle while para resolver un problema práctico relacionado con la construcción de una pirámide de bloques

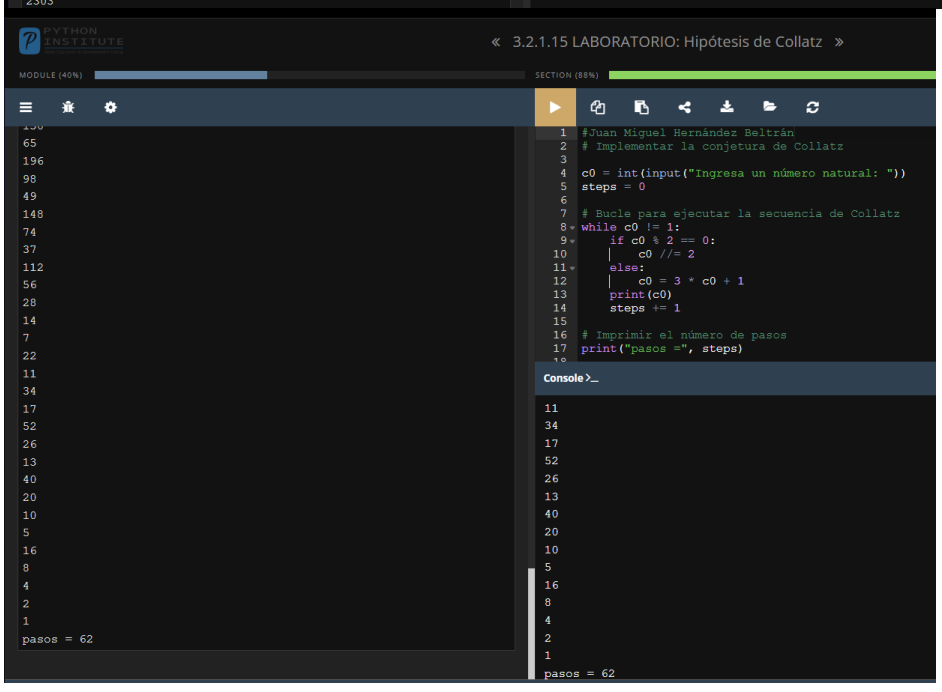


3.2.1.15 LABORATORIO: Hipótesis de Collatz

En esta actividad, aprendí a implementar la hipótesis de Collatz utilizando un bucle while al desarrollar un programa que toma un número natural y aplica las reglas de la hipótesis, pude reflejar un concepto matemático en código.

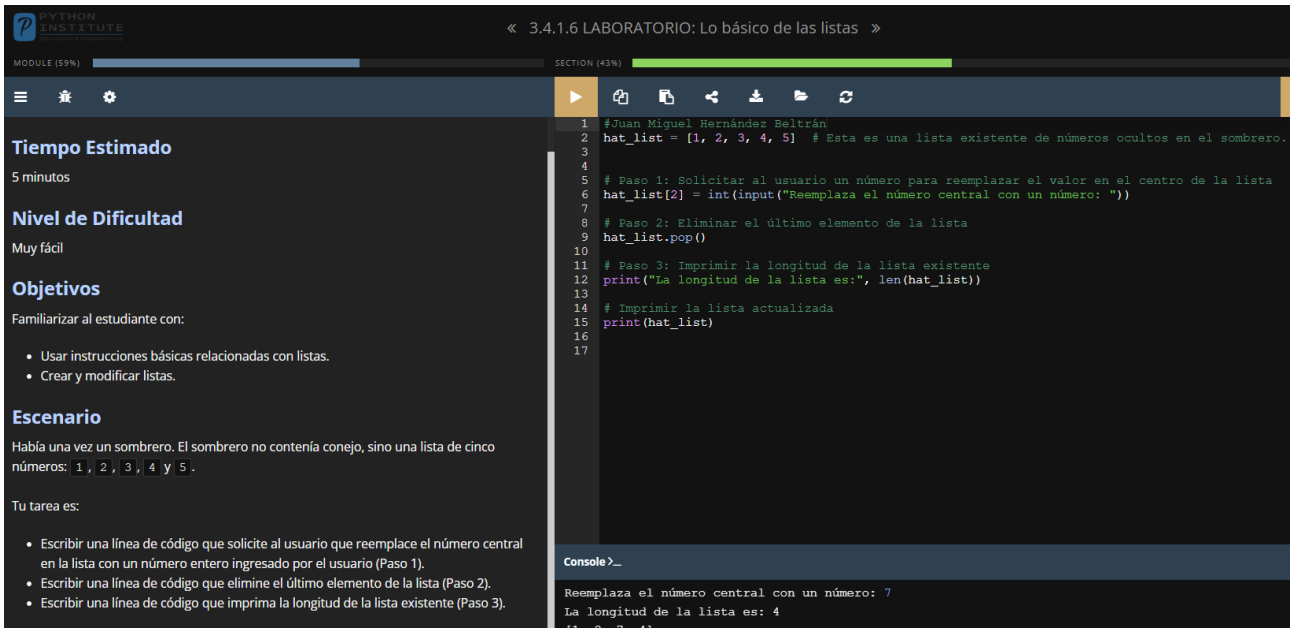

5
16
8
4
2
1
pasos = 17
Entrada de muestra: 16
Salida esperada:
8
4
2
1
pasos = 4
Entrada de muestra: 1023
Salida esperada:
3070
1535
4606
2303
6910
3455
#Juan Miguel Hernández Beltrán
# Implementar la conjetura de Collatz
c0 = int(input("Ingresa un número natural: "))
steps = 0
# Bucle para ejecutar la secuencia de Collatz
while c0 != 1:
if c0 % 2 == 0:
c0 //= 2
else:
c0 = 3 * c0 + 1
print(c0)
steps += 1
# Imprimir el número de pasos
print("pasos =", steps)
Console >_
Ingresa un número natural: 15
46
23
70
35
106
53
160
80
40
20
10
5
16
8
4

« 3.2.1.15 LABORATORIO: Hipótesis de Collatz »
MODULE (40%)
SECTION (88%)
1
pasos = 17
Entrada de muestra: 16
Salida esperada:
8
4
2
1
pasos = 4
Entrada de muestra: 1023
Salida esperada:
3070
1535
4606
2303
#Juan Miguel Hernández Beltrán
# Implementar la conjetura de Collatz
c0 = int(input("Ingresa un número natural: "))
steps = 0
# Bucle para ejecutar la secuencia de Collatz
while c0 != 1:
if c0 % 2 == 0:
c0 //= 2
else:
c0 = 3 * c0 + 1
print(c0)
steps += 1
# Imprimir el número de pasos
print("pasos =", steps)
Console >_
Ingresa un número natural: 16
8
4
2
1
pasos = 4


« 3.2.1.15 LABORATORIO: Hipótesis de Collatz »
MODULE (40%)
SECTION (88%)
65
196
98
49
148
74
37
112
56
28
14
7
22
11
34
17
52
26
13
40
20
10
5
16
8
4
2
1
pasos = 62
#Juan Miguel Hernández Beltrán
# Implementar la conjetura de Collatz
c0 = int(input("Ingresa un número natural: "))
steps = 0
# Bucle para ejecutar la secuencia de Collatz
while c0 != 1:
if c0 % 2 == 0:
c0 //= 2
else:
c0 = 3 * c0 + 1
print(c0)
steps += 1
# Imprimir el número de pasos
print("pasos =", steps)
Console >_
11
34
17
52
26
13
40
20
10
5
16
8
4
2
1
pasos = 62

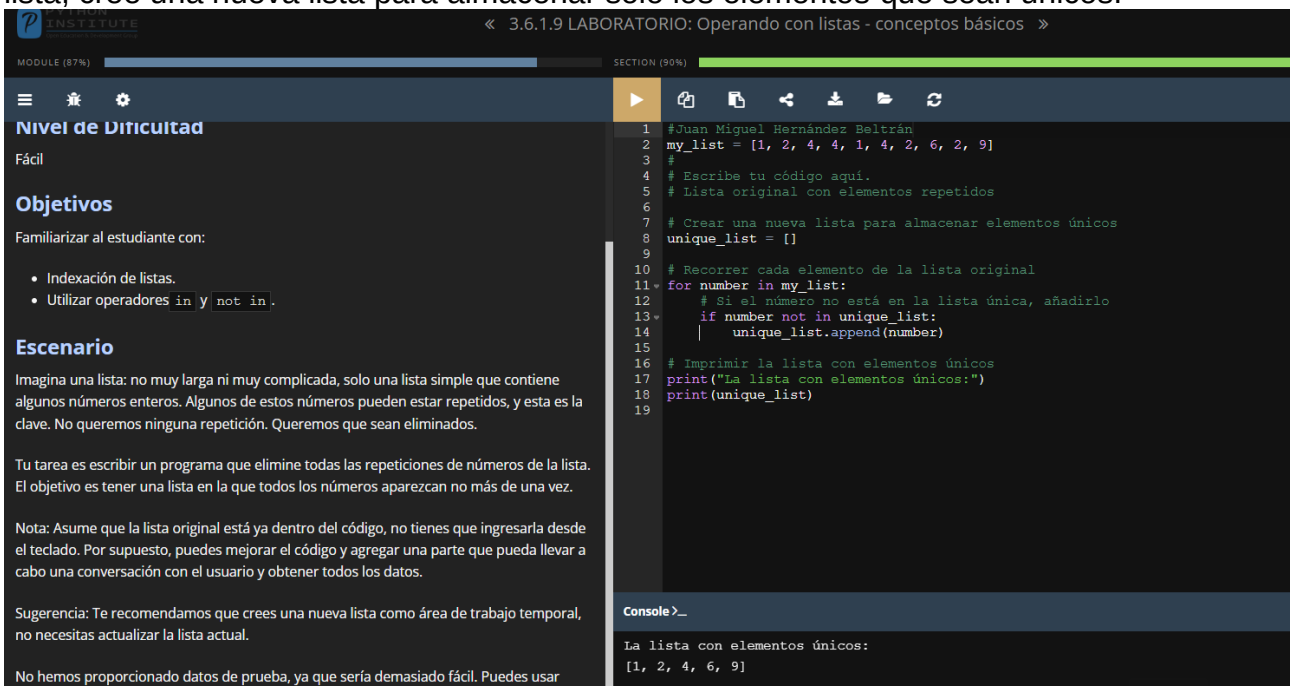
3.4.1.6 LABORATORIO: Lo básico de las listas

En esta actividad, aprendí a trabajar con listas lo cual me permitió crear y modificar estructuras de datos de manera mas sencilla.

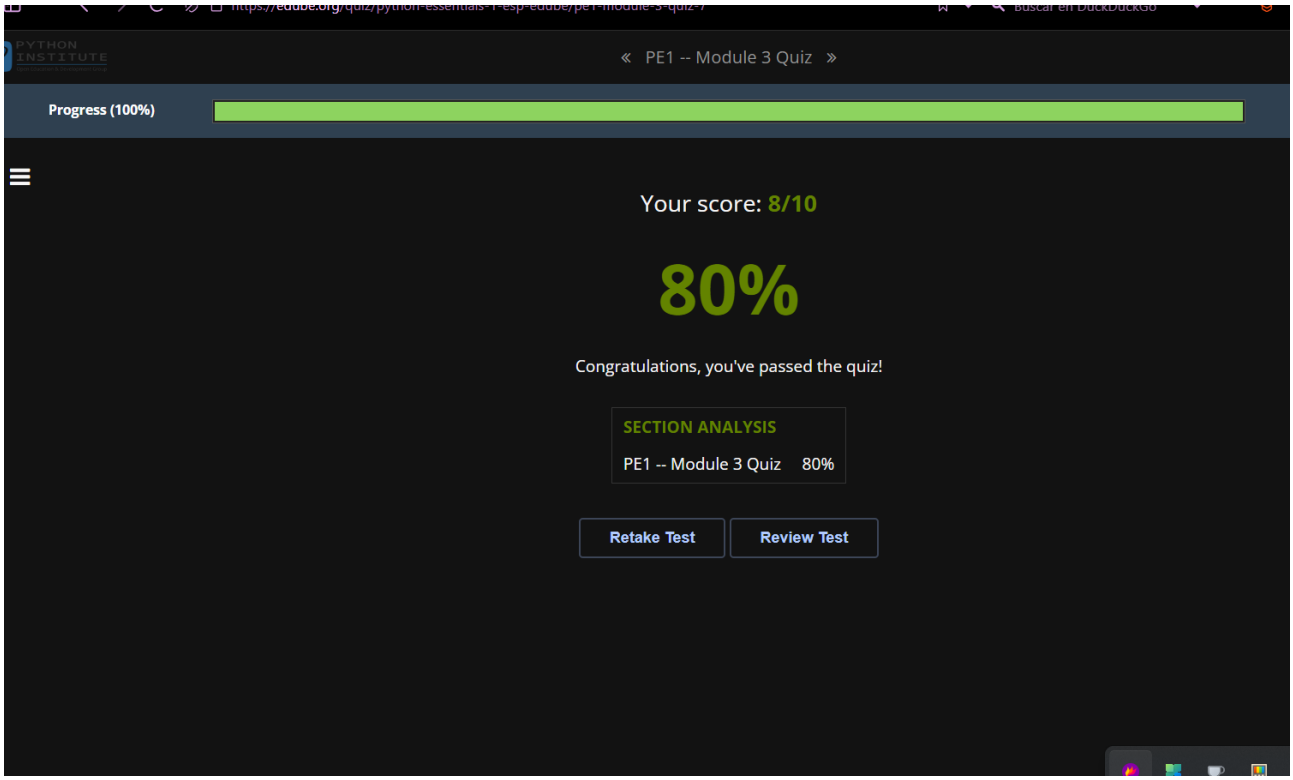


3.6.1.9 LABORATORIO: Operando con listas - conceptos básicos

En esta actividad, aprendí a trabajar con la indexación de listas y a utilizar los operadores in y not in Al desarrollar un programa que elimina las repeticiones de números en una lista, creé una nueva lista para almacenar solo los elementos que sean únicos.



Pantalla de Quiz



Pantalla de Examen Módulo 3