

Laboratorio 3

PYTHON INSTITUTE
Open Education & Development Group

« 2.1.1.19 LABORATORIO: Dando formato a la salida »

MODULE (28%)

SECTION (95%)

LABORATORIO

Tiempo Estimado

5-15 minutos

Nivel de Dificultad

Fácil

Objetivos

- Experimentar con el código Python existente.
- Descubrir y solucionar errores básicos de sintaxis.
- Familiarizarse con la función `print()` y sus capacidades de formato.

Escenario

Recomendamos que **juegues con el código** que hemos escrito para ti y que realices algunas correcciones (quizás incluso destructivas). Siéntete libre de modificar cualquier parte del código, pero hay una condición: aprende de tus errores y saca tus propias conclusiones.

Intenta:

- Minimizar el número de invocaciones de la función `print()` insertando la secuencia `\n` en las cadenas

Sandbox

```

1 print("Versión Original:")
2
3 print(" *")
4 print(" * *")
5 print(" * * *")
6 print(" * * * *")
7 print("**** *")
8 print(" * * *")
9 print(" * * *")
10 print(" * * * *")
11
12 print("Con menos invocaciones de 'print()': ")
13 #####
14 print(" * *\n * *\n * *\n * *\n * *\n *")
15 print(" * *\n * *\n * *\n *")
16
17 print("Más alto:")
18
19 print(" *")
20 print(" * *")
21 print(" * * *")
22 print(" * * *")
23 print(" * * *")
24 print(" * * *")
25 print(" * * *")
26 print(" * * *")
27 print("*****")

```

Console >_

```

Con menos invocaciones de 'print()':
*
* *
* * *
* * *
* * *

```

Laboratorio 4

PYTHON INSTITUTE
Open Education & Development Group

<

»

https://edube.org/learn/python-essentials-1-esp/laboratorio-literales-de-python-cadenas-2

« 2.2.1.11 LABORATORIO: Literales de Python - Cadenas »

MÓDULO (45%)SECTION (92%)

≡ ⚙️ 🏠

Tiempo Estimado

5-10 minutos

Nivel de Dificultad

Fácil

Objetivos

- Familiarizarse con la función `print()` y sus capacidades de formato.
- Practicar el codificar cadenas.
- Experimentar con el código de Python.

Escenario

Escribe una sola línea de código, utilizando la función `print()`, así como los caracteres de nueva línea y escape, para obtener la salida esperada de tres líneas.

Salida Esperada

"Estoy"
""aprendiendo""
"""Python"""

salida

▶ 🔍 ↻ ⬇ ⬆ ⌂

Sandbox

1 print(' "Estoy"\n', ""aprendiendo"", """python""")

Console ›_↺

"Estoy"
""aprendiendo""
"""python"""

Laboratorio 5

PYTHON INSTITUTE
Open Education & Development Group

MODULE (70%)

« 2.4.1.7 LABORATORIO: Variables »

SECTION (64%)

☰ ⚙️ 🔍

• Experimentar con el código en Python.

Escenario

A continuación una historia:

Érase una vez en la Tierra de las Manzanas, Juan tenía tres manzanas, María tenía cinco manzanas, y Adán tenía seis manzanas. Todos eran muy felices y vivieron por muchísimo tiempo. Fin de la Historia.

Tu tarea es:

- Crear las variables: `juan`, `maria`, y `adan`.
- Asignar valores a las variables. El valor debe de ser igual al número de manzanas que cada quien tenía.
- Una vez almacenados los números en las variables, imprimir las variables en una línea, y separar cada una de ellas con una coma.
- Después se debe crear una nueva variable llamada `total_manzanas` y se debe igualar a la suma de las tres variables anteriores.
- Imprime el valor almacenado en `total_manzanas` en la consola.
- Experimenta con tu código:** crea nuevas variables, asigna diferentes valores a ellas, y realiza varias operaciones aritméticas con ellas (por ejemplo, +, -, *, /, //, etc.). Intenta poner una cadena con un entero juntos en la misma línea, por ejemplo, "Número Total de Manzanas:" y `total_manzanas`.

▶ 📄 ↩ ⬇ 🗑 ↺

Sandbox

```
1 juan = 3
2 maria = 5
3 adan = 6
4
5 print("Manzanas de Juan: ", juan, "Manzanas de María: ", maria, "Manzanas de Adán: ", adan)
6
7 total_manzanas = juan + maria + adan
8
9 print("Número Total de Manzanas", total_manzanas)
10
```

Console ▶

Manzanas de Juan: 3 Manzanas de María: 5 Manzanas de Adán: 6
Número Total de Manzanas 14

Laboratorio 6

https://edube.org/learn/python-essentials-1-esp/laboratorio-variables-un-sencillo-conversidor-1

PYTHON INSTITUTE
Open Education & Development Group

<< 2.4.1.9 LABORATORIO: Variables, un sencillo conversidor >>

MODULE (73%)

SECTION (82%)

Sandbox

Hay una cosa interesante más que esta ocurriendo. ¿Puedes ver otra función dentro de la función `print()`? Es la función `round()`. Su trabajo es redondear la salida del resultado al número de decimales especificados en el paréntesis, y regresar un valor flotante (dentro de la función `round()` se puede encontrar el nombre de la variable, el nombre, una coma, y el número de decimales que se desean mostrar). Se hablará más de esta función muy pronto, no te preocupes si no todo queda muy claro. Solo se quiere impulsar tu curiosidad.

Después de completar el laboratorio, abre Sandbox, y experimenta más. Intenta escribir diferentes convertidores, por ejemplo, un convertidor de USD a EUR, un convertidor de temperatura, etc. ¡Deja que tu imaginación vuele! Intenta mostrar los resultados combinando cadenas y variables. Intenta utilizar y experimentar con la función `round()` para redondear tus resultados a uno, dos o tres decimales. Revisa que es lo que sucede si no se provee un dígito al redondear. Recuerda probar tus programas.

Experimenta, saca tus propias conclusiones, y aprende. Sé curioso.

Resultado Esperado

7.38 millas son 11.88 kilómetros
12.25 kilómetros son 7.61 millas

salida

1 kilometers = 12.25
2 miles = 7.38
3
4 miles_to_kilometers = miles * 1.61
5 kilometers_to_miles = kilometers / 1.61
6
7 print(miles, "millas son", round(miles_to_kilometers, 2), "kilómetros")
8 print(kilometers, "kilómetros son", round(kilometers_to_miles, 2), "millas")
9

Console>

7.38 millas son 11.88 kilómetros
12.25 kilómetros son 7.61 millas

Prev

Next

Laboratorio 7

[←](#) [Python Institute](#) [Open Education & Development Group](#)

LABORATORIO: Operadores y expresiones

MODULE (75%) SECTION (91%)

LABORATORIO

Tiempo Estimado

10-15 minutos

Nivel de Dificultad

Fácil

Objetivos

- Familiarizarse con los conceptos de números, operadores y operaciones aritméticas en Python.
- Realizar cálculos básicos.

Escenario

Observa el código en el editor: lee un valor `flotante`, lo coloca en una variable llamada `x`, e imprime el valor de la variable llamada `y`. Tu tarea es completar el código para evaluar la siguiente expresión:

$$3x^3 - 2x^2 + 3x - 1$$

El resultado debe ser asignado a `y`.

```

1 x = 
2 x = float(x)
3 y = 3 * x**3 - 2 * x**2 + 3 * x - 1
4 print("y =", y)
5
6 x = 1
7 x = float(x)
8 y = 3 * x**3 - 2 * x**2 + 3 * x - 1
9 print("y =", y)
10
11 x = -1
12 x = float(x)
13 y = 3 * x**3 - 2 * x**2 + 3 * x - 1
14 print("y =", y)
15

```

Console >_

```

y = -1.0
y = 3.0
y = -9.0

```

Prev Next

Laboratorio 8

PYTHON INSTITUTE

Open Education & Development Group

« 2.5.1.2 LABORATORIO: Comentarios »

MODULE (75%)

SECTION (67%)

☰

🔍

⚙️

LABORATORIO

Tiempo Estimado

5 minutos

Nivel de Dificultad

Muy Fácil

Objetivos

- Familiarizarse con el concepto de comentarios en Python.
- Utilizar y no utilizar los comentarios.
- Reemplazar los comentarios con código.
- Experimentar con el código de Python.

Escenario

El código en el editor contiene comentarios. Intenta mejorarlo: agrega o quita comentarios donde consideres que sea apropiado (en ocasiones el remover un comentario lo hace mas legible), además, cambia el nombre de las variables donde consideres que esto mejorará la comprensión del código.

NOTA

Los comentarios son muy importantes. No solo hacen que el programa sea más fácil de

▶

📄

🗑️

↶

↷

↺

Sandbox

```
1 # este programa calcula los segundos en cierto número de horas determinadas
2
3 a = 2 # número de horas
4 seconds = 3600 # número de segundos en una hora
5
6 print("Horas: ", a) #imprime el numero de horas
7 print("Segundos en Horas: ", a * seconds) # se imprime el numero de segundos en determinado numero de h
8
9 #este es el fin del programa que calcula el numero de segundos en 2 horas
10
```

Console ▶_

Horas: 2
Segundos en Horas: 7200

Laboratorio 9

PYTHON INSTITUTEOpen Source & Development Group

« 2.6.1.9 LABORATORIO: Entradas y salidas simples »

MODULE (93%)SECTION (69%)

☰🔍⚙️

Tiempo Estimado
5-10 minutos

Nivel de Dificultad
Fácil

Objetivos

- Familiarizarse con la entrada y salida de datos en Python.
- Evaluar expresiones simples.

Escenario

La tarea es completar el código para evaluar y mostrar el resultado de cuatro operaciones aritméticas básicas.

El resultado debe ser mostrado en consola.

Quizá no podrás proteger el código de un usuario que intente dividir entre cero. Por ahora, no hay que preocuparse por ello.

Prueba tu código - ¿Produce los resultados esperados?

No te mostraremos ningún dato de prueba, eso sería demasiado sencillo.

> 📄 ↶ ⤴ ⬇ ⬆ 🔁

Sandbox

```
1 # ingresa un valor flotante para la variable a aquí
2 flo = float(input("Ingresa un primer valor: "))
3
4 # ingresa un valor flotante para la variable b aquí
5 flo2 = float(input("Ingresa un segundo valor: "))
6
7 # muestra el resultado de la suma aquí
8 print(flo + flo2)
9 # muestra el resultado de la resta aquí
10 print(flo - flo2)
11 # muestra el resultado de la multiplicación aquí
12 print(flo * flo2)
13 # muestra el resultado de la división aquí
14 print(flo / flo2)
15
16 print("\n¡Eso es todo, amigos!")
17
```

Console >_ 🔁

Ingresar un primer valor: 10
Ingresar un segundo valor: 10


20.0
0.0
100.0
1.0

¡Eso es todo, amigos!

Laboratorio 10

[illegible]

Laboratorio 11



« 2.6.1.11 LABORATORIO: Operadores y expresiones »

MODULE (96%) SECTION (85%)

Datos de Prueba

Entrada de muestra:

```
12
17
59
```

Salida esperada: 13:16

Entrada de muestra:

```
23
58
642
```

Salida esperada: 10:40

Entrada de muestra:

```
0
1
2939
```

Salida esperada: 1:0


```
1 hour = int(input("Hora de inicio (horas): "))
2 mins = int(input("Minuto de inicio (minutos): "))
3 dura = int(input("Duración del evento (minutos): "))
4 mins = mins + dura # encuentra el número total de minutos
5 hour = hour + mins // 60
6 mins = mins % 60
7 hour = hour % 24
8 print(hour, ":", mins, sep='')
9
```

Console >_

```
Hora de inicio (horas): 12
Minuto de inicio (minutos): 17
Duración del evento (minutos): 59
13:16
Hora de inicio (horas): 23
Minuto de inicio (minutos): 58
Duración del evento (minutos): 642
10:40
Hora de inicio (horas): 0
Minuto de inicio (minutos): 1
Duración del evento (minutos): 2939
1:0
```

Sandbox

Laboratorio 12



« 3.1.1.4 LABORATORIO: Preguntas y respuestas »

MODULE (7%) SECTION (33%)

Resultado esperado: False

Ejemplo de entrada: 99

Resultado esperado: False

Ejemplo de entrada: 100

Resultado esperado: True

Ejemplo de entrada: 101

Resultado esperado: True

Ejemplo de entrada: -5

Resultado esperado: False

Ejemplo de entrada: +123

Resultado esperado: True

```
1 n = int(input("Ingrese el valor de n: "))
2 print(n >= 100)
3
```

Console >_

```
Ingrese el valor de n: 55
False
Ingrese el valor de n: 99
False
Ingrese el valor de n: 100
True
Ingrese el valor de n: 101
True
Ingrese el valor de n: -5
False
Ingrese el valor de n: +123
True
```

Sandbox

Laboratorio 13

PYTHON INSTITUTE
Open Source & Development Group

3.1.1.10 LABORATORIO: Operadores de comparación y ejecución condicional »

MODULE (15%)

SECTION (73%)

Menu

Icons

Sandbox

"espatifilo".

- Imprima "¡ESPATIFILIO!, ¡No [entrada]!" de lo contrario. Nota: [entrada] es la cadena que se toma como entrada.

Prueba tu código con los datos que te proporcionamos. ¡Y hazte de un ESPATIFILIO también!

Datos de Prueba

Entrada de muestra: `espatifilo`

Resultado esperado: `No, ¡quiero un gran ESPATIFILIO!`

Entrada de ejemplo: `pelargonio`

Resultado esperado: `¡ESPATIFILIO!, ¡No pelargonio!`

Entrada de muestra: `ESPATIFILIO`

Resultado esperado: `Si, ¡El ESPATIFILIO es la mejor planta de todos los tiempos!`

```

1 entrada = input("Ingrese la palabra: ")
2
3 if entrada == "ESPATIFILIO":
4     print("Si, ¡El ESPATIFILIO! es la mejor planta de todos los tiempos!")
5 elif entrada == "espatifilo":
6     print("No, ¡quiero un gran ESPATIFILIO!")
7 else:
8     print("¡ESPATIFILIO!, ¡No " + entrada + "!")
9

```

Console >_

```

Ingrese la palabra: espatifilo
No, ¡quiero un gran ESPATIFILIO!
Ingrese la palabra: pelarfonio
¡ESPATIFILIO!, ¡No pelarfonio!
Ingrese la palabra:
Ingrese la palabra: ESPATIFILIO
Si, ¡El ESPATIFILIO! es la mejor planta de todos los tiempos!

```

Laboratorio 14

PYTHON INSTITUTE
Open Source & Development Group

3.1.1.11 LABORATORIO: Fundamentos de la sentencia if-else »

MODULE (16%)

SECTION (80%)

☰

🔍

⚙️

▶

📄

🔗

👤

📁

🔄

Sandbox

Observa el código en el editor, súbelo, dale un valor de entrada y genera un resultado, por lo que debes completarlo con algunos cálculos inteligentes.

Prueba tu código con los datos que hemos proporcionado.

Datos de Prueba

Entrada de muestra:

Resultado esperado:

Entrada de muestra:

Resultado esperado:

Entrada de muestra:

Resultado esperado:

Entrada de muestra:

Resultado esperado:

```

1 ingreso = float(input("Introduce el ingreso anual: "))
2
3- if ingreso < 85528:
4     impuesto = ingreso * 0.18 - 556.02
5- else:
6     impuesto = (ingreso - 85528) * 0.32 + 14839.02
7
8- if impuesto < 0.0:
9     impuesto = 0.0
10
11 impuesto = round(impuesto)
12 print("El impuesto es:", impuesto, "pesos")
13

```

Console >_

```

Introduce el ingreso anual: 10000
El impuesto es: 1244 pesos
Introduce el ingreso anual: 100000
El impuesto es: 19470 pesos
Introduce el ingreso anual: 1000
El impuesto es: 0 pesos
Introduce el ingreso anual: -100
El impuesto es: 0 pesos

```

Laboratorio 15

PYTHON INSTITUTE
Open Education & Development Group

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

MODULE (18%)SECTION (87%)

Sandbox

Datos de Prueba

Entrada de muestra:

2000

Resultado esperado:

Año Bisiesto

Entrada de muestra:

2015

Resultado esperado:

Año Común

Entrada de muestra:

1999

Resultado esperado:

Año Común

Entrada de muestra:

1996

Resultado esperado:

Año Bisiesto

Entrada de muestra:

1580

Resultado esperado:

No esta dentro del periodo del calendario Gregoriano

```
1 año = int(input("Introduce un año: "))
2
3 if año < 1582:
4     print("No esta dentro del periodo del calendario Gregoriano")
5 else:
6     if año % 4 != 0:
7         print("Año Común")
8     elif año % 100 != 0:
9         print("Año Bisiesto")
10    elif año % 400 != 0:
11        print("Año Común")
12    else:
13        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 esta dentro del periodo del calendario Gregoriano

PrevNext

Laboratorio 16

PYTHON INSTITUTE
Open Education & Development Group

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

MODULE (24%)

SECTION (18%)

adivinar 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"`.

¡El mago está contando contigo! No lo decepciones.

Sandbox

```

1 # Completa el código siguiente para adivinar el número secreto
2
3 +=====+
4 |
5 | +-----+
6 | | ¡Bienvenido a mi juego, muggle!  |
7 | | Introduce un número entero     |
8 | | y adivina qué número he        |
9 | | elegido para ti.                 |
10 | | Entonces,                       |
11 | | ¿Cuál es el número secreto?    |
12 | +-----+
13 | """)
14
15 num = int(input(" Ingresa el número: "))
16
17 while num != secret_number:
18     print("¡Ja, ja! ¡Estás atrapado en mi bucle!")
19     num = int(input(" Ingresa el número: "))
20 print("¡Bien hecho, muggle! Eres libre ahora")

```

Console>_

```

1 |
2 | elegido para ti. |
3 | Entonces,       |
4 | ¿Cuál es el número secreto? |
5 | +=====+
6 |
7 | Ingresa el número: 100
8 | ¡Ja, ja! ¡Estás atrapado en mi bucle!
9 | Ingresa el número: 200
10 | ¡Ja, ja! ¡Estás atrapado en mi bucle!
11 | Ingresa el número: 2540
12 | ¡Ja, ja! ¡Estás atrapado en mi bucle!
13 | Ingresa el número: 777
14 | ¡Bien hecho, muggle! Eres libre ahora

```

INFO EXTRA

Por cierto, observa la función `print()`. La forma en que lo hemos utilizado aquí se llama *impresión multilinea*. Puede utilizar **comillas triples** para imprimir cadenas en varias líneas para facilitar la lectura del texto o crear un diseño especial basado en texto. Experimenta con ello.

Laboratorio 17

PYTHON INSTITUTEOpen Education & Development Group

« 3.2.1.6 LABORATORIO: Fundamentos del bucle for: el conteo »

MODULE (28%)

SECTION (35%)

≡ ⚙️ 🔍

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!`".

Utiliza el esqueleto que hemos proporcionado en el editor.

INFO EXTRA

Ten en cuenta que el código en el editor contiene dos elementos que pueden no ser del todo claros en este momento: la sentencia `import time` y el método `sleep()`. Vamos a hablar de ellos pronto.

Por el momento, nos gustaría que supieras que hemos importado el módulo `time` y hemos utilizado el método `sleep()` para suspender la ejecución de cada función posterior de `print()` dentro del bucle `for` durante un segundo, de modo que el mensaje enviado a la consola se parezca a un conteo real. No te preocupes, pronto aprenderás más sobre módulos y métodos.

Salida esperada

```
1 Mississippi
2 Mississippi
3 Mississippi
4 Mississippi
5 Mississippi
¡Listos o no, ahí voy!
```

>

```
1 import time
2
3 for segundo in range(1, 6):
4     print(segundo, "Mississippi")
5     time.sleep(1)
6
7 print("¡Listos o no, ahí voy!")
8 
```

Sandbox

Console >_

```
1 Mississippi
2 Mississippi
3 Mississippi
4 Mississippi
5 Mississippi
¡Listos o no, ahí voy!
```

Laboratorio 18

Python Institute

Open Education & Development Group

« 3.2.1.9 LABORATORIO: La sentencia break - Atascado en un bucle »

MODULE (32%)

SECTION (53%)

LABORATORIO

Tiempo Estimado
10 minutos

Nivel de Dificultad
Fácil

Objetivos
Familiarizar al estudiante con:

- Utilizar la instrucción `break` en los bucles.
- Reflejar situaciones de la vida real en código de computadora.

Escenario
La instrucción `break` se implementa para salir/terminar un bucle.

Diseña un programa que use un bucle `while` y le pida continuamente al usuario que ingrese una palabra a menos que ingrese "chupacabra" como la palabra de salida secreta, en cuyo caso el mensaje ";Has dejado el bucle con éxito". Debe imprimirse en la pantalla y el bucle debe terminar.

Sandbox

```
1 while True:  
2     palabra = input(";Estás atrapado en un bucle infinito!\nIngresa la palabra secreta para dejar el bucle ")  
3     if palabra == "chupacabra":  
4         break  
5     print(";Has dejado el bucle con éxito!")  
6
```

Console
_>
;Estás atrapado en un bucle infinito!
Ingresa la palabra secreta para dejar el bucle: hola
;Estás atrapado en un bucle infinito!
Ingresa la palabra secreta para dejar el bucle: nooo
;Estás atrapado en un bucle infinito!
Ingresa la palabra secreta para dejar el bucle: estoy atrapado
;Estás atrapado en un bucle infinito!
Ingresa la palabra secreta para dejar el bucle:
;Estás atrapado en un bucle infinito!
Ingresa la palabra secreta para dejar el bucle: chupacabra
;Has dejado el bucle con éxito!

Laboratorio 19

← ↻ 🔒 <https://edube.org/learn/python-essentials-1-esp/laboratorio-la-sentencia-continue-el-feo-devorador-de-vocales-1> 🔊 ☆ 🔄 📄 🌐 🏠 ... 🔍

PYTHON INSTITUTE
Open Education & Development Group

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

MODULE (33%) SECTION (59%)

Salida esperada:

```
G
R
G
R
Y
```

Entrada de muestra:

Salida esperada:

```
B
S
T
M
S
```

Entrada de muestra:

Salida esperada:

```
1 user_word = input("Ingresa tu palabra: ")
2 user_word = user_word.upper()
3
4 for letra in user_word:
5     if letra == "A":
6         continue
7     elif letra == "E":
8         continue
9     elif letra == "I":
10        continue
11    elif letra == "O":
12        continue
13    elif letra == "U":
14        continue
15    else:
16        print(letra)
17
```

Console >_

```
Ingresa tu palabra: Gregory
G
R
G
R
Y
Ingresa tu palabra: abstemious
B
S
T
M
S
Ingresa tu palabra: iouea
```

Sandbox

Laboratorio 20

← ↻ 🔒 <https://edube.org/learn/python-essentials-1-esp/laboratorio-la-sentencia-continue-el-bonito-devorador-de-vocales> 🔊 ☆ 🔄 📄 🌐 🏠 ... 🔍

PYTHON INSTITUTE
Open Education & Development Group

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

MODULE (35%) SECTION (65%)

Prueba tu programa con los datos que le proporcionamos.

Datos de Prueba

Entrada de muestra:

Salida esperada:

```
GRGRY
```

Entrada de muestra:

Salida esperada:

```
BSTMS
```

Entrada de muestra:

Salida esperada:

```
1 word_without_vowels = ""
2
3 user_word = input("Ingresa tu palabra: ")
4 user_word = user_word.upper()
5
6 for letra in user_word:
7     if letra == "A":
8         continue
9     elif letra == "E":
10        continue
11    elif letra == "I":
12        continue
13    elif letra == "O":
14        continue
15    elif letra == "U":
16        continue
17    else:
18        word_without_vowels += letra
19
20 print(word_without_vowels)
21
```

Console >_

```
Ingresa tu palabra: Gregory
GRGRY
Ingresa tu palabra: abstemious
BSTMS
Ingresa tu palabra: iouea
```

Sandbox

Laboratorio 21

← ↻ 🔒 <https://edube.org/learn/python-essentials-1-esp/laboratorio-fundamentos-del-bucle-while> A ☆ ⌂ 🌐 🏠 ...

PYTHON INSTITUTE
Open Education & Development Group

« 3.2.1.14 LABORATORIO: Fundamentos del bucle while »

MODULE (39%) SECTION (82%)

⌵ ⌶ ⚙

▶ 📄 📁 🔍 ⬅ ➡ 🔄

Sandbox

Prueba tu código con los datos que hemos proporcionado.

Datos de Prueba

Entrada de muestra:

Salida esperada:

Entrada de muestra:

Salida esperada:

Entrada de muestra:

Salida esperada:

Entrada de muestra:

Salida esperada:

```
1 blocks = int(input("Ingresa el número de bloques: "))
2
3 height = 0
4 in_layer = 1
5 while in_layer <= blocks:
6     height += 1
7     blocks -= in_layer
8     in_layer += 1
9
10 print("La altura de la pirámide:", height)
11
```

Console >_

```
Ingresa el número de bloques: 6
La altura de la pirámide: 3
Ingresa el número de bloques: 20
La altura de la pirámide: 5
Ingresa el número de bloques: 1000
La altura de la pirámide: 44
Ingresa el número de bloques: 2
La altura de la pirámide: 1
```

Laboratorio 22

← ↻ 🔒 <https://edube.org/learn/python-essentials-1-esp/laboratorio-hip-oacute-tesis-de-collatz-1> A ☆ ⌂ 🌐 🏠 ...

PYTHON INSTITUTE
Open Education & Development Group

« 3.2.1.15 LABORATORIO: Hipótesis de Collatz »

MODULE (40%) SECTION (88%)

⌵ ⌶ ⚙

▶ 📄 📁 🔍 ⬅ ➡ 🔄

Sandbox

Test Data

Entrada de muestra:

Salida esperada:

```
46
23
70
35
106
53
160
80
40
20
10
5
16
8
4
2
1
pasos = 17
```

```
1 c0 = int(input("Ingresa c0: "))
2
3 if c0 > 1:
4     steps = 0
5     while c0 != 1:
6         if c0 % 2 != 0:
7             cnew = 3 * c0 + 1
8         else:
9             cnew = c0 // 2
10        print(c0)
11        c0 = cnew
12        steps += 1
13    print("pasos =", steps)
14 else:
15    print("Valor de c0 incorrecto")
16
```

Console >_

```
35
106
53
160
80
40
20
10
5
16
8
4
2
pasos = 17
```

Prev Next

Laboratorio 23

←

↻

https://edube.org/learn/python-essentials-1-esp/laboratorio-lo-b-aacute-sico-de-las-listas-1

⌵

☆

🔄

📖

🔍

👤

⋮

🔗

PYTHON INSTITUTE

Open Education & Development Group

MODULE (59%)

« 3.4.1.6 LABORATORIO: Lo básico de las listas »

SECTION (43%)

☰

🔍

⚙️

▶

📄

📁

🔗

📥

📤

🔄

Sandbox

LABORATORIO

Tiempo Estimado

5 minutos

Nivel de Dificultad

Muy fácil

Objetivos

Familiarizar al estudiante con:

- Usar instrucciones básicas relacionadas con listas.
- Crear y modificar listas.

Escenario

Había una vez un sombrero. El sombrero no contenía conejo, sino una lista de cinco números: 1, 2, 3, 4 y 5.

Tu tarea es:

- Escribir una línea de código que solicite al usuario que reemplace el número central en la lista con un número entero ingresado por el usuario (Paso 1).
- Escribir una línea de código que elimine el último elemento de la lista (Paso 2).

```
1 hat_list = [1, 2, 3, 4, 5]
2
3 # Paso 1
4 hat_list[2] = int(input("Ingresa un número entero: "))
5
6 # Paso 2
7 del hat_list[-1]
8
9 # Paso 3
10 print(len(hat_list))
11
```

Console >_

Ingresar un número entero: 10
4

Laboratorio 24

[Python Institute](#)

3.4.1.13 LABORATORIO: Lo básico de las listas - Los Beatles »

MODULE (65%) SECTION (93%)

la banda más vendida en la historia. Algunas personas los consideran el acto más influyente de la era del rock. De hecho, se incluyeron en la compilación de la revista Time de las 100 personas más influyentes del siglo XX.

a banda sufrió muchos cambios de formación, que culminaron en 1962 con la formación de John Lennon, Paul McCartney, George Harrison y Richard Starkey (mejor conocido como Ringo Starr).

Escribe un programa que refleje estos cambios y le permita practicar con el concepto de listas. Tu tarea es:

- Paso 1: Crea una lista vacía llamada `beatles`.
- Paso 2: Emplea el método `append()` para agregar los siguientes miembros de la banda a la lista: John Lennon, Paul McCartney y George Harrison.
- Paso 3: Emplea el bucle `for` y el `append()` para pedirle al usuario que agregue los siguientes miembros de la banda a la lista: Stu Sutcliffe, y Pete Best.
- Paso 4: Usa la instrucción `del` para eliminar a Stu Sutcliffe y Pete Best de la lista.
- Paso 5: Usa el método `insert()` para agregar a Ringo Starr al principio de la lista.

Por cierto, ¿eres fan de los Beatles? (Los Beatles son una de las bandas favoritas de Greg. Pero espera...¿Quién es Greg?)

```

1 # paso 1:
2 Beatles = []
3 print("Paso 1:", Beatles)
4
5 # paso 2:
6
7 Beatles.append("John Lennon")
8 Beatles.append("Paul McCartney")
9 Beatles.append("George Harrison")
10 print("Paso 2:", Beatles)
11
12 # paso 3:
13 for members in range(2):
14     Beatles.append(input("Nuevo miembro de la banda: "))
15 print("Paso 3:", Beatles)
16
17 # paso 4:
18 del Beatles[-1]
          
```

Console>

```

Paso 1: []
Paso 2: ['John Lennon', 'Paul McCartney', 'George Harrison']
Nuevo miembro de la banda: Stu Sutcliffe
Nuevo miembro de la banda: Pete Best
Paso 3: ['John Lennon', 'Paul McCartney', 'George Harrison', 'Stu Sutcliffe', 'Pete Best']
Paso 4: ['John Lennon', 'Paul McCartney', 'George Harrison']
Paso 5: ['RingoStarr', 'John Lennon', 'Paul McCartney', 'George Harrison']
Los Fav: 4
          
```

Laboratorio 25

<https://edube.org/learn/python-essentials-1-esp/laboratorio-operando-con-listas-conceptos-b-aacute-sicos-1>

« 3.6.1.9 LABORATORIO: Operando con listas - conceptos básicos »

MODULE (88%) SECTION (90%)

Sandbox

LABORATORIO

Tiempo Estimado

10-15 minutos

Nivel de Dificultad

Fácil

Objetivos

Familiarizar al estudiante con:

- Indexación de listas.
- Utilizar operadores `in` y `not in`.

Escenario

Imagina una lista: no muy larga ni muy complicada, solo una lista simple que contiene algunos números enteros. Algunos de estos números pueden estar repetidos, y esta es la clave. No queremos ninguna repetición. Queremos que sean eliminados.

Tu tarea es escribir un programa que elimine todas las repeticiones de números de la lista. El objetivo es tener una lista en la que todos los números aparezcan no más de una vez.

```

1 my_list = [1, 2, 4, 4, 1, 4, 2, 6, 2, 9]
2 new_list = []
3 for num in my_list:
4     if num not in new_list:
5         new_list.append(num)
6 my_list = new_list[:]
7 print("La lista con elementos únicos:")
8 print(my_list)
9 
```

Console>_

```

La lista con elementos únicos:
[1, 2, 4, 6, 9]

```

Laboratorio 26