¡Felicitaciones! ¡Aprobaste!

Calificación recibida 100 % **Para Aprobar** 80 % o más

Ir al siguiente elemento

Estás viendo una versión traducida automáticamente de esta evaluación
Puedes volver a ver el contenido en su idioma original si lo prefieres. No perderás el progreso que hayas conseguido si cambias el idioma.
Mostrar la versión en Inglés
Desestimar

1. ¿En qué se diferencian los bucles `while` y `for` en Python? 1 / 1 punto

- `while` pueden utilizarse con todos los tipos de datos; los bucles `for` sólo pueden utilizarse con números.
- `for` pueden anidarse, pero los bucles `while` no.
- (x) `while` los bucles iteran mientras una condición es verdadera; `for` los bucles iteran a través de una secuencia de elementos.
- `while` los bucles se pueden interrumpir utilizando break; los bucles `for` se interrumpen utilizando continue.

Correcto
Entendido Utilice los bucles `while` cuando desee que su código se ejecute repetidamente mientras una condición sea verdadera, y los bucles `for` cuando desee ejecutar un bloque de código por cada elemento de una secuencia.

2. ¿Qué opción solucionaría este bucle `for` para imprimir los números 12, 18, 24, 30, 36? 1 / 1 punto

```
1  for n in range(6,18,3):
2    print(n*2)
```

- ```
  1  for n in range(6,18,3):
  2    print(n+2)
  ```
- (x) ```
  1 for n in range(6,18+1,3):
  2   print(n*2)
  ```
- ```
  1  for n in range(12,36,6):
  2    print(n*2)
  ```
- ```
  1  for n in range(0,36+1,6):
  2    print(n)
  ```

Correcto
¡Buen trabajo! Para incluir 18 en el rango, añádale 1. El segundo parámetro podría escribirse como 18+1 o 19.

3. ¿Qué bucles de `for` imprimirán todos los números pares del 0 al 18? Seleccione todos los que correspondan. 1 / 1 punto

- [x] ```
  1 for n in range(19):
  2   if n % 2 == 0:
  3   print(n)
  ```
- [ ] ```
  1  for n in range(18+1):
  2    print(n**2)
  ```
- [ ] ```
  1 for n in range(0,18+1,2):
  2   print(n*2)
  ```
- [x] ```
  2    print(n+n)
  ```

Correcto
Correcto Este bucle imprimirá todos los números pares del 0 al 18. El rango de "n" empezará en 0 y terminará en 18 (se excluye el valor de rango final de 19). La variable "n" se incrementará por defecto en 1 en cada iteración del bucle. La sentencia `if` utiliza el operador `modulo` (%) para comprobar si la variable "n" es divisible por 2. Si es cierto, la sentencia `if` imprimirá el valor de "n" y saldrá de nuevo al bucle `for` para la siguiente iteración de "n"

4. Rellene los espacios en blanco para que el bucle `for` imprima los 10 primeros números cúbicos (x**3) en un rango que empiece con x=1 y termine con x=10. 1 / 1 punto

```
1  for x in range(1,11):
2    print(x**3)
```

Ejecutar
Restablecer

```
1
8
27
64
125
216
343
512
729
1000
```

Correcto
¡Lo has clavado! Has conseguido el código para imprimir los 10 primeros cubos.

5. Escriba un bucle `for` con una función `range()` de tres parámetros que imprima los múltiplos de 7 entre 0 y 100. Imprima un múltiplo por línea y evite imprimir cualquier número que no sea múltiplo de 7. Recuerde que 0 también es múltiplo de 7. 1 / 1 punto

```
1  for n in range(0, 101, 7):
2    print(n)
```

Ejecutar
Restablecer

```
0
7
14
21
28
35
42
49
56
63
70
77
84
91
98
```

Correcto
¡Impresionante! Está haciendo que Python haga todo el trabajo por usted.

6. ¿Cuál de estas opciones daría salida sólo a las vocales de la siguiente cadena? Seleccione todas las que correspondan. 1 / 1 punto

```
1  input = "Four score and seven years ago"
```

- [x] ```
  1 for c in input:
  2   if c.lower() in ['a', 'e', 'i', 'o', 'u']:
  3   print(c)
  ```
- [x] ```
  1  print([c for c in input if c.lower() in ['a', 'e', 'i', 'o', 'u']])
  ```

Correcto
Correcto Puede utilizar un bucle `for` para examinar cada carácter de la cadena. Observe que el uso de la función `lower()` le permite encontrar tanto las vocales mayúsculas como las minúsculas.

Correcto
Correcto Aquí puede utilizar una comprensión de lista para reunir sólo los caracteres que coincidan con la expresión condicional.

- [ ] ```
  1 print(input.count("aeiou"))
  ```
- [ ] ```
  1  for c in range(len(input)):
  2    if c in ['a', 'e', 'i', 'o', 'u']:
  3      print(c)
  ```

7. ¿Cuál de estas afirmaciones es cierta sobre el troceado de cadenas? 1 / 1 punto

- El método `slice()` puede utilizarse para trocear una cadena.
- Si el índice inicial es negativo, Python genera un error de ejecución.
- (x) Si el índice inicial es negativo, Python cuenta hacia atrás desde el final de la cadena.
- Al trocear una cadena, debe proporcionar siempre los índices inicial y final.

Correcto
Correcto Puede utilizar índices negativos para extraer rápidamente un trozo del final de una cadena.