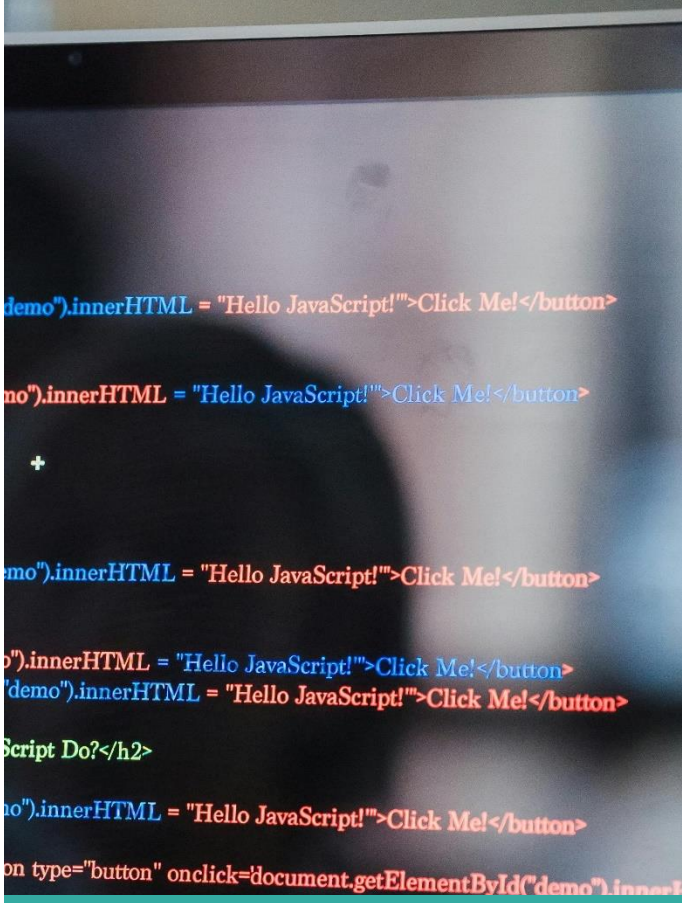
Lección 003



17 OCTUBRE

ROLOTECH

Creado por: Morales Campuzano Jose Rodrigo

EstructurasControl01.py

Código

```
# If

i = 8
if (i == 8):
   if (i < 100):
      print("i es menor a 100")
   elif (i < 10):
      print("i es menor a 10")
   else:
      print("i es mayor a 100")
```

Explicación

Este código utiliza una estructura condicional if para evaluar el valor de i:

1. Se define i con el valor 8.
2. La primera condición if (i == 8) es verdadera, por lo que se ejecuta el bloque correspondiente.
    - Dentro de este bloque, se evalúa otra condición:
        - if (i < 100): Esta condición también es verdadera, por lo que se imprime: **"i es menor a 100"**.
    - Las siguientes condiciones (elif y else) no se evalúan porque la primera condición verdadera ha sido ejecutada.

En resumen, el código verifica que i es 8 y que es menor que 100, imprimiendo el mensaje correspondiente.

Salída

```
$ python estructurasControl01.py
i es menor a 100
```

EstructurasControl02.py

Código

```
# If

i = 8
if (i < 100):
   print("i es menor a 100")
if (i < 10):
   print("i es menor a 10")
else:
   print("i es mayor a 100")
```

Explicación

Este código utiliza estructuras condicionales if y else para evaluar el valor de i:

1. Se define i con el valor 8.
2. La primera condición if (i < 100) se evalúa:
    - Es verdadera, por lo que se imprime: **"i es menor a 100"**.
3. La segunda condición if (i < 10) se evalúa:
    - Es verdadera (ya que 8 es menor que 10), por lo que se imprime: **"i es menor a 10"**.
4. La cláusula else no se ejecuta porque la condición anterior fue verdadera.

En resumen, el código verifica que i es 8, imprime que es menor a 100, y también que es menor a 10. La parte del else no se ejecuta en este caso.

Salída

```
$ python estructurasControl02.py
i es menor a 100
i es menor a 10]
```

EstructurasControl03.py

Código

```
# If
```

```
def comparar(i):
    if (i > 0):
        return 'Positivo'
    elif (i < 0):
        return 'Negativo'
    else:
        return 'Cero'

lambda_comparar1 = lambda i: {i>0: 'Positivo', i <0: 'Negativo'}.get(True, "Cero")

lambda_comparar2 = lambda i: 'Positivo' if i > 0 else ('Negativo' if i<0 else 'Cero')


for i in range(-2,3,1):
    print(f'Def: {i} es {comparar(i)}')


for i in range(-2,3,1):
    print(f'Lambda1: {i} es {lambda_comparar1(i)}')

for i in range(-2,3,1):
    print(f'Lambda2: {i} es {lambda_comparar2(i)}')
```

Explicación

ste código define una función y dos expresiones lambda para clasificar un número como positivo, negativo o cero, y luego las evalúa en un rango de números. Aquí está el desglose:

1. **Función comparar(i)**:
    - Toma un argumento i.
    - Retorna 'Positivo' si i es mayor que 0.
    - Retorna 'Negativo' si i es menor que 0.
    - Retorna 'Cero' si i es igual a 0.
2. **Lambda lambda_comparar1**:
    - Esta expresión lambda utiliza un diccionario para clasificar el número.
    - Si i > 0, devuelve 'Positivo'; si i < 0, devuelve 'Negativo'. Si ninguna de estas condiciones es verdadera, usa get(True, "Cero") para devolver 'Cero'.
3. **Lambda lambda_comparar2**:

- Esta expresión lambda utiliza una expresión condicional para clasificar el número.
- Retorna 'Positivo' si i es mayor que 0, 'Negativo' si i es menor que 0, y 'Cero' si no se cumple ninguna de estas condiciones.

4. **Bucle for**:
   - Se itera sobre el rango de números de -2 a 2 (inclusive) con un paso de 1.
   - En cada iteración, se imprimen los resultados de cada método de comparación (función y lambdas):
     - Para cada i, se imprime su clasificación utilizando la función comparar.
     - Luego, se imprime su clasificación usando lambda_comparar1.
     - Finalmente, se imprime su clasificación usando lambda_comparar2.

Salída

```
$ python estructurasControl03.py
Def: -2 es Negativo
Def: -1 es Negativo
Def: 0 es Cero
Def: 1 es Positivo
Def: 2 es Positivo
Lambda1: -2 es Negativo
Lambda1: -1 es Negativo
Lambda1: 0 es Cero
Lambda1: 1 es Positivo
Lambda1: 2 es Positivo
Lambda2: -2 es Negativo
Lambda2: -1 es Negativo
Lambda2: 0 es Cero
Lambda2: 1 es Positivo
Lambda2: 2 es Positivo
```

EstructurasControl04.py

Código

```
# For
```

```
cadena = "Hola Mundo!"
for c in cadena:
    print(c)
```

Explicación

Este código utiliza un bucle for para iterar sobre cada carácter de la cadena cadena:

1. **Definición de la cadena**: Se asigna la cadena "Hola Mundo!" a la variable cadena.
2. **Bucle for**:
    - El bucle recorre cada carácter de la cadena.
    - En cada iteración, se imprime el carácter actual c.

Salída

```
$ python estructurasControl04.py
H
o
l
a

M
u
n
d
o
!
```

EstructurasControl05.py

Código

```
# For

for c in range(0, 11):
    print(c)
```

Explicación

Este código utiliza un bucle for para iterar sobre un rango de números:

1. **Bucle for**:
    - range(0, 11) genera una secuencia de números desde 0 hasta 10 (el límite superior 11 no está incluido).
    - En cada iteración, el valor actual de la variable c se imprime.

Salída

```
$ python estructurasControl05.py
0
1
2
3
4
5
6
7
8
9
10
```

EstructurasControl06.py

Código

```
# For

print('Del 3 al 15 de 3 en 3')
for c in range(3, 15, 3):
    print(c)


print('Del 10 al -5 de 4 en 4')
for c in range(10, -6, -4):
    print(c)
```

Explicación

Este código utiliza bucles for con la función range() para iterar sobre dos secuencias de números con diferentes pasos:

1. **Primera sección**:
    - **Mensaje**: Imprime el mensaje "Del 3 al 15 de 3 en 3".
    - **Bucle for**: range(3, 15, 3) genera una secuencia de números desde 3 hasta 14 (el límite superior 15 no está incluido) con un paso de 3.
    - **Salida**: Imprime los números 3, 6, 9, 12, y 15.
2. **Segunda sección**:
    - **Mensaje**: Imprime el mensaje "Del 10 al -5 de 4 en 4".
    - **Bucle for**: range(10, -6, -4) genera una secuencia de números desde 10 hasta -5 (el límite inferior -6 no está incluido) con un paso de -4.
    - **Salida**: Imprime los números 10, 6, 2, y -2.

Salída

```
$ python estructurasControl06.py
Del 3 al 15 de 3 en 3
3
6
9
12
Del 10 al -5 de 4 en 4
10
6
2
-2
```

EstructurasControl07.py

Código

```
# For

lista = ['gato', 'perro', 'raton']

for posicion, elemento in enumerate(lista):
    print(posicion, elemento)
```

Explicación

Este código utiliza un bucle for junto con la función enumerate() para iterar sobre una lista y obtener tanto el índice como el elemento correspondiente en cada iteración:

1. **Definición de la lista**: Se asigna la lista ['gato', 'perro', 'raton'] a la variable lista.
2. **Bucle for**:
    - enumerate(lista) genera pares de índice y elemento para cada elemento de la lista.
    - En cada iteración, posicion toma el valor del índice y elemento toma el valor del elemento correspondiente en la lista.
3. **Salida**: Se imprime la posicion y el elemento en cada iteración.

Salída

```
$ python estructurasControl07.py
0 gato
1 perro
2 raton
```

EstructurasControl08.py

Código

```
# For anidado

for i in range(1, 4):
    for j in range(1, 4):
        for k in range(1, 4):
            for l in range(1, 4):
                print(f'i={i} j={j} k={k} l={l}')
```

Explicación

Este código utiliza bucles for anidados para crear combinaciones de valores en un rango específico. Aquí está el desglose:

1. **Bucle exterior**: for i in range(1, 4)
    - Itera i desde 1 hasta 3 (el límite superior 4 no está incluido).
2. **Segundo bucle**: for j in range(1, 4)

- Por cada valor de i, itera j desde 1 hasta 3.

3. **Tercer bucle**: for k in range(1, 4)
   - Por cada combinación de i y j, itera k desde 1 hasta 3.
4. **Cuarto bucle**: for l in range(1, 4)
   - Por cada combinación de i, j y k, itera l desde 1 hasta 3.
5. **Salida**: En cada iteración del bucle más interno, se imprime el valor actual de i, j, k y l.

Salída

```
$ python estructurasControl08.py
i=1 j=1 k=1 l=1
i=1 j=1 k=1 l=2
i=1 j=1 k=1 l=3
i=1 j=1 k=2 l=1
i=1 j=1 k=2 l=2
i=1 j=1 k=2 l=3
i=1 j=1 k=3 l=1
i=1 j=1 k=3 l=2
i=1 j=1 k=3 l=3
i=1 j=2 k=1 l=1
i=1 j=2 k=1 l=2
i=1 j=2 k=1 l=3
i=1 j=2 k=2 l=1
i=1 j=2 k=2 l=2
i=1 j=2 k=2 l=3
i=1 j=2 k=3 l=1
i=1 j=2 k=3 l=2
i=1 j=2 k=3 l=3
i=1 j=3 k=1 l=1
i=1 j=3 k=1 l=2
i=1 j=3 k=1 l=3
i=1 j=3 k=2 l=1
i=1 j=3 k=2 l=2
i=1 j=3 k=2 l=3
i=1 j=3 k=3 l=1
i=1 j=3 k=3 l=2
i=1 j=3 k=3 l=3
i=2 j=1 k=1 l=1
i=2 j=1 k=1 l=2
i=2 j=1 k=1 l=3
i=2 j=1 k=2 l=1
```

```
i=2 j=1 k=2 l=2
i=2 j=1 k=2 l=3
i=2 j=1 k=3 l=1
i=2 j=1 k=3 l=2
i=2 j=1 k=3 l=3
i=2 j=2 k=1 l=1
i=2 j=2 k=1 l=2
i=2 j=2 k=1 l=3
i=2 j=2 k=2 l=1
i=2 j=2 k=2 l=2
i=2 j=2 k=2 l=3
i=2 j=2 k=3 l=1
i=2 j=2 k=3 l=2
i=2 j=2 k=3 l=3
i=2 j=3 k=1 l=1
i=2 j=3 k=1 l=2
i=2 j=3 k=1 l=3
i=2 j=3 k=2 l=1
i=2 j=3 k=2 l=2
i=2 j=3 k=2 l=3
i=2 j=3 k=3 l=1
i=2 j=3 k=3 l=2
i=2 j=3 k=3 l=3
i=3 j=1 k=1 l=1
i=3 j=1 k=1 l=2
i=3 j=1 k=1 l=3
i=3 j=1 k=2 l=1
i=3 j=1 k=2 l=2
i=3 j=1 k=2 l=3
i=3 j=1 k=3 l=1
i=3 j=1 k=3 l=2
i=3 j=1 k=3 l=3
i=3 j=2 k=1 l=1
i=3 j=2 k=1 l=2
i=3 j=2 k=1 l=3
i=3 j=2 k=2 l=1
i=3 j=2 k=2 l=2
i=3 j=2 k=2 l=3
i=3 j=2 k=3 l=1
i=3 j=2 k=3 l=2
i=3 j=2 k=3 l=3
i=3 j=3 k=1 l=1
i=3 j=3 k=1 l=2
i=3 j=3 k=1 l=3
i=3 j=3 k=2 l=1
```

```
i=3 j=3 k=2 l=2
i=3 j=3 k=2 l=3
i=3 j=3 k=3 l=1
i=3 j=3 k=3 l=2
i=3 j=3 k=3 l=3
```

EstructurasControl09.py

Código

```
# For
import random

numeros1 = [x for x in range(5)]
print(numeros1)

numeros2 = [x for x in range(2,7,2)]
print(numeros2)

numeros3 = [random.randint(1,10) for _ in range(5)]
print(numeros3)
```

Explicación

Este código utiliza comprensión de listas para crear tres listas de números, mostrando diferentes formas de generar elementos en cada lista. Aquí está el desglose:

1. **Importación de módulo**: import random
    - Importa el módulo random para generar números aleatorios.
2. **Creación de numeros1**:
    - numeros1 = [x for x in range(5)] crea una lista de números enteros desde 0 hasta 4 (el límite superior 5 no está incluido).
    - **Salida**: numeros1 contendrá [0, 1, 2, 3, 4].
3. **Creación de numeros2**:
    - numeros2 = [x for x in range(2, 7, 2)] crea una lista de números enteros comenzando desde 2 hasta 6 (el límite superior 7 no está incluido) con un paso de 2.
    - **Salida**: numeros2 contendrá [2, 4, 6].

4. **Creación de numeros3**:
    - numeros3 = [random.randint(1, 10) for _ in range(5)] genera una lista de 5 números enteros aleatorios, cada uno en el rango de 1 a 10 (ambos extremos incluidos).
    - **Salida**: numeros3 contendrá 5 números aleatorios, por ejemplo, [3, 7, 1, 9, 4], pero los valores específicos cambiarán cada vez que se ejecute el código

Salída

```
$ python estructurasControl09.py
[0, 1, 2, 3, 4]
[2, 4, 6]
[5, 9, 9, 1, 3]
```

EstructurasControl10.py

Código

```
# For

animales = ['Leon', 'Zebra', 'Murcielago', 'Humano']
comidas = ['Carnivoro', 'Herbivoro', 'Insectivoro', 'Omnivoro']

for animal, comida in zip(animales, comidas):
    print(f'{animal} es {comida}')
```

Explicación

Este código utiliza un bucle for junto con la función zip() para iterar simultáneamente sobre dos listas, animales y comidas. Aquí está el desglose:

1. **Definición de listas**:
    - animales contiene ['Leon', 'Zebra', 'Murcielago', 'Humano'].
    - comidas contiene ['Carnivoro', 'Herbivoro', 'Insectivoro', 'Omnivoro'].
2. **Bucle for con zip()**:
    - for animal, comida in zip(animales, comidas): crea pares de elementos de las listas animales y comidas, emparejando el primer elemento de animales con el primer de comidas, el segundo con el segundo, y así sucesivamente.

- En cada iteración, animal toma el valor de un animal de la lista y comida toma el valor correspondiente de la lista de comidas.

3. **Salida**: En cada iteración, se imprime una frase que relaciona el animal con su tipo de comida.

Salída

```
$ python estructurasControl10.py
Leon es Carnivoro
Zebra es Herbivoro
Murcielago es Insectivoro
Humano es Omnivoro
```

EstructurasControl11.py

Código

```
# For

cadena = 'Murcielago'

for c in cadena:
    if c == 'l':
        print('g', end='')
    elif c == 'g':
        print('l', end='')
    else:
        print(c, end='')
```

Explicación

Este código itera sobre los caracteres de la cadena cadena (que contiene 'Murcielago') y realiza ciertas modificaciones en función de los caracteres que encuentra. Aquí está el desglose:

1. **Definición de la cadena**:
    - cadena = 'Murcielago' es la cadena que se va a procesar.
2. **Bucle for**:
    - for c in cadena: itera sobre cada carácter c en la cadena.

3. **Condiciones dentro del bucle**:
    - if c == 'l': verifica si el carácter actual es 'l'. Si es así, imprime 'g' en su lugar.
    - elif c == 'g': verifica si el carácter actual es 'g'. Si es así, imprime 'l' en su lugar.
    - else: para todos los demás caracteres, imprime el carácter original.
4. **Uso de end=''**:
    - print(..., end='') se utiliza para evitar el salto de línea después de cada impresión, lo que permite que todos los caracteres se impriman en la misma línea.

Salída

```
$ python estructurasControl11.py
Murciegalo
```

EstructurasControl12.py

Código

```
# For

for i in range(1, 5):
    print(i)
else:
    print(f"Termino {i}")
```

Explicación

Este código utiliza un bucle for junto con una cláusula else. Aquí está el desglose:

1. **Bucle for**:
    - for i in range(1, 5): itera sobre los números del 1 al 4 (inclusive), ya que range(1, 5) genera la secuencia [1, 2, 3, 4].
2. **Impresión de cada número**:
    - Dentro del bucle, print(i) imprime el valor actual de i en cada iteración.
3. **Cláusula else**:

- La cláusula else asociada al bucle se ejecuta después de que el bucle for ha terminado todas sus iteraciones sin interrupción (es decir, no se ha utilizado un break para salir del bucle prematuramente).
- En este caso, print(f"Termino {i}") imprime un mensaje que indica que el bucle ha terminado, y muestra el último valor de i que fue 4.

Salída

```
$ python estructurasControl12.py
1
2
3
4
Termino 4
```