Introducción a la Programación Segura

Estructuras de decisión, bucles y funciones del lenguaje Python.

INTRODUCCIÓN A LA PROGRAMACIÓN SEGURA

UNIDAD 2: ESTRUCTURAS DE DECISIÓN, BUCLES Y FUNCIONES DEL LENGUAJE PYTHON.

Contenidos

1. Operadores relacionales.
2. Condiciones y ejecución condicional.
3. Declaración if, if-else, elif.
4. Estructuras de control: Bucles y condicionales.
5. Operadores lógicos.
6. Funciones con retorno, envío de valores, parámetros y argumentos.

UNIDAD DE APRENDIZAJE

UNIDAD N°2: ESTRUCTURAS DE DECISIÓN, BUCLES Y FUNCIONES DEL LENGUAJE PYTHON.

Aprendizaje Esperado:

2.1. Desarrolla scripts de Python, para entregar solución a problemas de mediana complejidad, considerando los estándares internacionales de programación y la evaluación de los resultados obtenidos en base a la evidencia.

Haciendo preguntas y recibiendo respuestas con operadores relacionales:

Operadores relacionales :

Los operadores relacionales son símbolos que se utilizan para realizar comparaciones entre valores en la programación y obtener un resultado verdadero o falso (también conocido como valor booleano). Estos operadores te permiten hacer preguntas sobre las relaciones entre diferentes variables o valores.

Operadores Relacionales		
==	Igualdad	Comprueba si dos valores son iguales.
!=	Desigualdad	Comprueba si dos valores no son iguales.
>	Mayor que	Comprueba si el valor de la izquierda es mayor que el valor de la derecha.
<	Menor que	Comprueba si el valor de la izquierda es menor que el valor de la derecha.
>=	Mayor o igual que	Comprueba si el valor de la izquierda es mayor o igual que el valor de la derecha.
<=	Menor o igual que	Comprueba si el valor de la izquierda es menor o igual que el valor de la derecha.

Haciendo preguntas y recibiendo respuestas con operadores relacionales:

Ejemplos:

```
x = 8
y = 12

# Preguntas y respuestas con operadores relacionales
print(x == y)   # False, ya que 8 no es igual a 12
print(x != y)   # True, ya que 8 no es igual a 12
print(x > y)    # False, ya que 8 no es mayor que 12
print(x >= y)   # False, ya que 8 no es mayor o igual que 12
print(x < y)    # True, ya que 8 es menor que 12
print(x <= y)   # True, ya que 8 es menor o igual que 12
```

```
False
True
False
False
True
True
```

Haciendo preguntas y recibiendo respuestas con operadores relacionales:

```python
# Ejemplo con operadores aritméticos, de comparación y lógicos
a = 5
b = 10
c = 15

# Expresión que combina diferentes operadores
resultado = (a + b < c) and (a * 2 >= b) or (c / 3 != a)

# Análisis paso a paso:
# 1. Suma: a + b = 5 + 10 = 15
# 2. Comparación: 15 < c -> 15 < 15 -> False (ya que 15 no es menor que 15)
# 3. Multiplicación: a * 2 = 5 * 2 = 10
# 4. Comparación: 10 >= b -> 10 >= 10 -> True (ya que 10 es igual a 10)
# 5. Operador lógico AND: False and True -> False (una de las condiciones es falsa)
# 6. División: c / 3 = 15 / 3 = 5.0 (resultado es un valor flotante)
# 7. Comparación: 5.0 != a -> 5.0 != 5 -> False (ya que 5.0 es igual a 5)
# 8. Operador lógico OR: False or False -> False (ambas condiciones son falsas)

print(resultado)  # Resultado final: False
```

```
False
```

Condiciones y ejecución condicional:

¿Qué es una condición en Python?

En Python, una condición es una expresión que evalúa a un valor booleano, es decir, a True o False. Las condiciones se utilizan en las estructuras de control como las declaraciones if, if-else y elif para tomar decisiones y ejecutar diferentes bloques de código según el resultado de la evaluación.

La Ejecución condicional:

La ejecución condicional es una característica esencial en la programación que permite tomar decisiones y ejecutar diferentes bloques de código según ciertas condiciones. Esto se logra mediante el uso de declaraciones if, if-else y la cláusula elif (abreviatura de "else if" en inglés). Estas estructuras de control son muy comunes en muchos lenguajes de programación y permiten que un programa se comporte de manera más inteligente y adaptable

La Declaración if:

La declaración if se utiliza para ejecutar un bloque de código solo si una condición específica es verdadera. Si la condición es verdadera, el bloque de código dentro del if se ejecuta, y si la condición es falsa, el bloque se omite y el programa continúa con la siguiente instrucción fuera del if.

if **condición** **:**

bloque de código si la condición es verdadera

```
if edad >= 18:
    print('Eres mayor de edad')
```

La Declaración if:

```
edad = int(input('Ingresa tu edad: '))

if edad >= 18:
    print('Eres mayor de edad')

print('Programa terminado.')
```

```
Ingresa tu edad: 23
Eres mayor de edad
Programa terminado.
```

La **condición** es **Verdadera**, por tanto, se **ejecuta** el **bloque** dentro del if.

```
Ingresa tu edad: 18
Eres mayor de edad
Programa terminado.
```

La **condición** es **Verdadera**, por tanto, se **ejecuta** el **bloque** dentro del if.

```
Ingresa tu edad: 15
Programa terminado.
```

La **condición** es **Falsa**, por tanto, **omite** el **bloque** dentro del if.

python™

Inacap

La Declaración if-else:

La declaración if-else se utiliza para ejecutar un bloque de código si la condición es verdadera y otro bloque si la condición es falsa. En este caso, solo uno de los bloques se ejecutará, dependiendo del resultado de la condición.

```
if condición :
    bloque de código si la condición es verdadera
else :
    bloque de código si la condición es falsa
```

La Declaración if-else:

```python
edad = int(input('Ingresa tu edad: '))

if edad >= 18:
    print('Eres mayor de edad')
else:
    print('Eres menor de edad')

print('Programa terminado.')
```

```
Ingresa tu edad: 23
Eres mayor de edad
Programa terminado.
```

La **condición** es **Verdadera**, por tanto, se **ejecuta** el **bloque** dentro del if.

```
Ingresa tu edad: 18
Eres mayor de edad
Programa terminado.
```

La **condición** es **Verdadera**, por tanto, se **ejecuta** el **bloque** dentro del if.

```
Ingresa tu edad: 15
Eres menor de edad
Programa terminado.
```

La **condición** es **Falsa**, por tanto, se ejecuta **bloque** dentro del else.

La Cláusula elif:

La cláusula elif se utiliza para evaluar múltiples condiciones en secuencia. Si la condición en el if es falsa, se evalúa la siguiente condición en la cláusula elif, y así sucesivamente. Si alguna de las condiciones es verdadera, se ejecuta el bloque de código correspondiente y el programa salta el resto de las cláusulas elif y el else.

```
if Condición 1 :
    bloque de código si la condición 1 es verdadera
elif Condición 2 :
    bloque de código si la condición 2 es verdadera
elif Condición 3 :
    bloque de código si la condición 3 es verdadera
else :
    bloque de código si las condiciones son falsas
```

La Declaración if-else:

```python
edad = int(input('Ingresa tu edad: '))

if edad >= 65:
    print('Eres anciano')
elif edad >= 18:
    print('Eres adulto')
elif edad >= 12:
    print('Eres adolescente')
else:
    print('Eres un niño')

print('Programa terminado.')
```

```
Ingresa tu edad: 70
Eres anciano
Programa terminado.
```

```
Ingresa tu edad: 45
Eres adulto
Programa terminado.
```

```
Ingresa tu edad: 16
Eres adolescente
Programa terminado.
```

```
Ingresa tu edad: 10
Eres un niño
Programa terminado.
```

Ejemplos:

1. Desarrolla un script que solicite el ingreso de un número entero positivo y muestre un mensaje por pantalla indicando si es un número par o impar:

```python
numero = float(input('Ingresa un número entero positivo: '))

if numero % 2 == 0:
    print("El número es par.")
else:
    print("El número es impar.")
```

```
Ingresa un número entero positivo: 8
El número es par.
```

```
Ingresa un número entero positivo: 15
El número es impar.
```

Ejemplos:

2. Desarrolla un script que solicite el ingreso de una calificación de un alumno y muestre por pantalla como resultado:
    - ¡Felicitaciones, estas eximido!, cuando la calificación sea igual o superior a 6,0.
    - ¡Felicidades, aprobaste!, cuando la calificación sea igual o superior a 4,0 pero menor a 6,0.
    - ¡Lo siento, reprobaste!, cuando la calificación sea menor a 4,0.

```python
nota = float(input('Ingresa tu nota: '))

if nota >= 6.0:
    print('¡Felicitaciones, estas eximido!')
elif nota >= 4.0:
    print('¡Felicidades, aprobaste!')
else:
    print('¡Lo siento, reprobaste!')
```

```
Ingresa tu nota: 5.2
¡Felicidades, aprobaste!
```

```
Ingresa tu nota: 6.4
¡Felicitaciones, estas eximido!
```

```
Ingresa tu nota: 3.5
¡Lo siento, reprobaste!
```

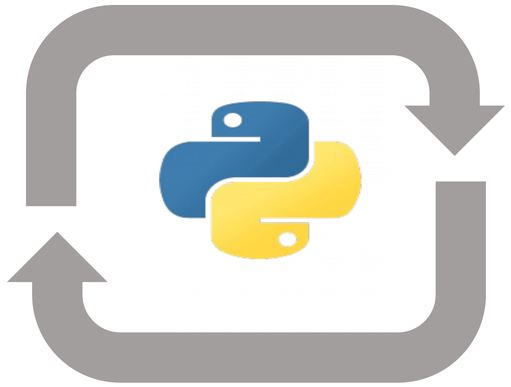
python™

Inacap

Los Bucles:

¿Qué los Bucles en Python?

Los bucles se utilizan para repetir un bloque de código múltiples veces, hasta que se cumpla una condición específica. Los bucles más comunes en Python son el bucle while y el bucle for.



Los Bucles:

El Bucle while:

El bucle while ejecuta un bloque de código siempre que una condición sea verdadera. La condición se verifica antes de cada iteración, y si es verdadera, el bloque de código se ejecuta. Si la condición se vuelve falsa en algún momento, el bucle se detiene y el programa continúa con la siguiente instrucción después del bucle.

```
while Condición :
    Bloque de código si la condición es verdadera
Instrucciones fuera del Bucle
```

Los Bucles:

El Bucle while:

El bucle while ejecuta un bloque de código siempre que una condición sea verdadera. La condición se verifica antes de cada iteración, y si es verdadera, el bloque de código se ejecuta. Si la condición se vuelve falsa en algún momento, el bucle se detiene y el programa continúa con la siguiente instrucción después del bucle.

```
contador = 1

while contador <= 10:
    print("Iteración {}".format(contador))
    contador += 1

print('Programa terminado.')
```

```
Iteración 1
Iteración 2
Iteración 3
Iteración 4
Iteración 5
Iteración 6
Iteración 7
Iteración 8
Iteración 9
Iteración 10
Programa terminado.
```

Los Bucles:

El Bucle for:

El bucle for se utiliza para iterar sobre una secuencia (como una lista, una tupla o una cadena) o un rango de valores específico. Cada elemento de la secuencia se toma uno a uno y se procesa en el bloque de código.

```
for elemento in iterable :
    Bloque de código que procesa al
    elemento actual de la secuencia
Instrucciones fuera del Bucle
```

Los Bucles:

El Bucle for:

El bucle for se utiliza para iterar sobre una secuencia (como una lista, una tupla o una cadena) o un rango de valores específico. Cada elemento de la secuencia se toma uno a uno y se procesa en el bloque de código.

```
frutas = ["Pera","Manzana", "Naranja", "Plátano"]

for fruta in frutas:
    print(fruta)


print('Programa terminado.')
```

```
Pera
Manzana
Naranja
Plátano
Programa terminado.
```

Los Bucles:

El Bucle for:

El bucle for se utiliza para iterar sobre una secuencia (como una lista, una tupla o una cadena) o un rango de valores específico. Cada elemento de la secuencia se toma uno a uno y se procesa en el bloque de código.

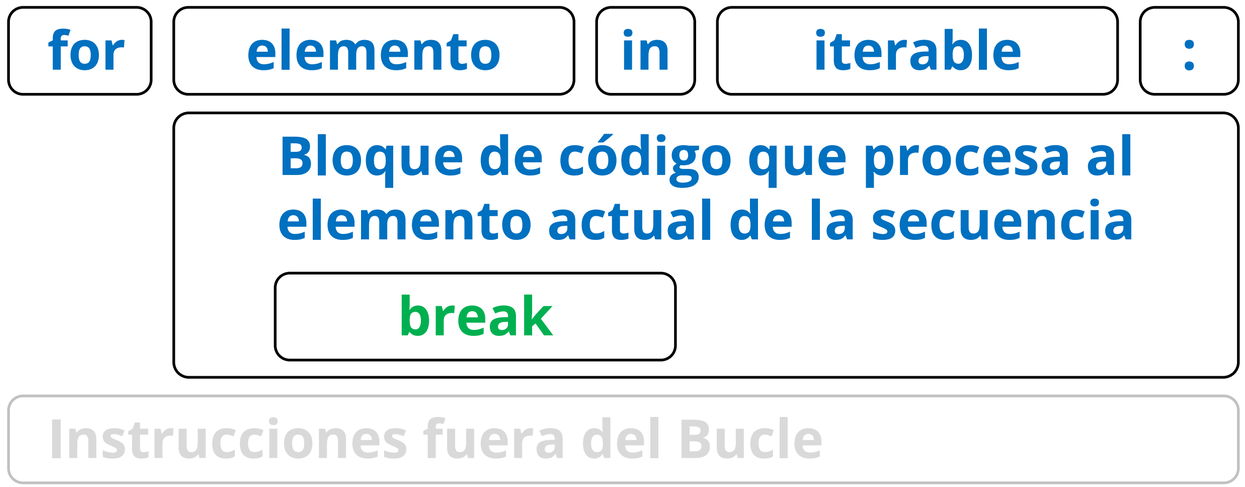
```
nombre = input('Ingrese su nombre: ')

for letra in nombre:
    print(letra)


print('Programa terminado.')
```

```
Ingrese su nombre: Teresa
T
e
r
e
s
a
Programa terminado.
```

Los Bucles:

Sentencia break:

La sentencia break se utiliza para salir del bucle inmediatamente cuando se cumple cierta condición. Es útil cuando necesitas detener el bucle antes de que se complete su ciclo normal.



Los Bucles:

Sentencia break:

La sentencia break se utiliza para salir del bucle inmediatamente cuando se cumple cierta condición. Es útil cuando necesitas detener el bucle antes de que se complete su ciclo normal.

```
frutas = ["Pera","Manzana", "Naranja", "Plátano"]
contador = 0

for fruta in frutas:
    if fruta == "Naranja":
        print("¡Encontre la",fruta,"!")
        break

    contador += 1
    print("Iteración",contador)
    print(fruta)

print('Programa terminado.')
```

```
Iteración 1
Pera
Iteración 2
Manzana
¡Encontre la Naranja !
Programa terminado.
```

```
frutas = ["Pera","Manzana", "Naranja", "Plátano"]
contador = 0

for fruta in frutas:
    if fruta == "Piña":
        print("¡Encontre la",fruta,"!")
        break

    contador += 1
    print("Iteración",contador)
    print(fruta)

print('Programa terminado.')
```

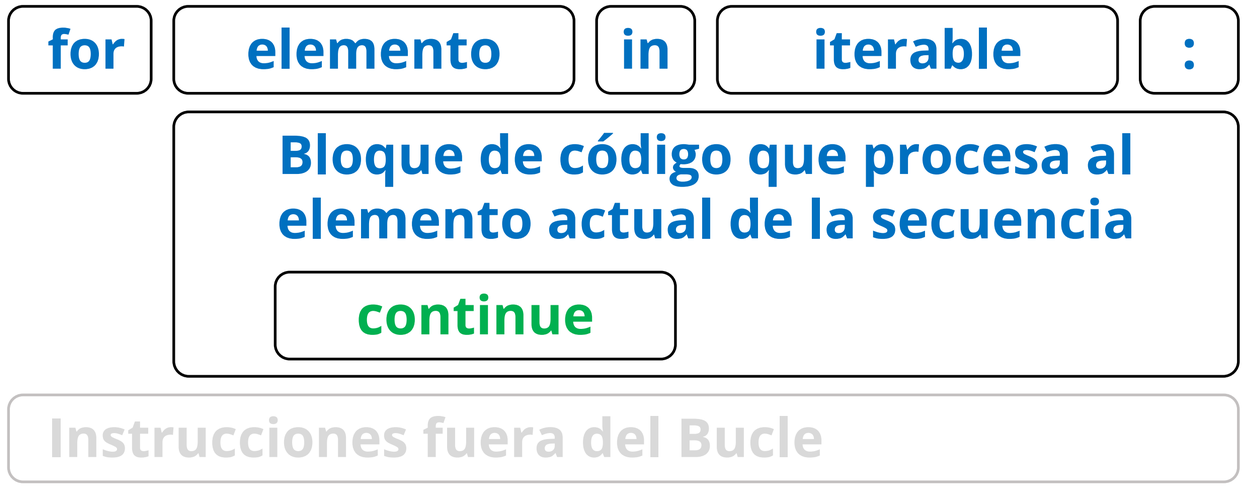
```
Iteración 1
Pera
Iteración 2
Manzana
Iteración 3
Naranja
Iteración 4
Plátano
Programa terminado.
```

La sentencia **break detiene el bucle**

Los Bucles:

Sentencia continue:

La sentencia continue se utiliza para saltar a la siguiente iteración del bucle, ignorando el resto del bloque de código que sigue a la sentencia continue. Es útil cuando necesitas omitir ciertas iteraciones basadas en una condición.



Los Bucles:

Sentencia continue:

La sentencia continue se utiliza para saltar a la siguiente iteración del bucle, ignorando el resto del bloque de código que sigue a la sentencia continue. Es útil cuando necesitas omitir ciertas iteraciones basadas en una condición.

```
frutas = ["Pera","Manzana", "Naranja", "Plátano"]
contador = 0

for fruta in frutas:
    if fruta == "Naranja":
        print("¡Encontre la",fruta,"!")
        continue

    contador += 1
    print("Iteración",contador)
    print(fruta)

print('Programa terminado.')
```

```
Iteración 1
Pera
Iteración 2
Manzana
¡Encontre la Naranja !
Iteración 3
Plátano
Programa terminado.
```

```
frutas = ["Pera","Manzana", "Naranja", "Plátano"]
contador = 0

for fruta in frutas:
    if fruta == "Piña":
        print("¡Encontre la",fruta,"!")
        continue

    contador += 1
    print("Iteración",contador)
    print(fruta)

print('Programa terminado.')
```

```
Iteración 1
Pera
Iteración 2
Manzana
Iteración 3
Naranja
Iteración 4
Plátano
Programa terminado.
```

La sentencia **continue detiene la iteración actual**

Lógica de la computadora y sus operadores:

La lógica de la computadora se refiere a la forma en que una computadora procesa y evalúa información utilizando operadores lógicos para tomar decisiones basadas en el valor de verdad de las expresiones. Los operadores lógicos son herramientas que permiten combinar o manipular valores booleanos (True o False) para obtener nuevos resultados. En Python, los operadores lógicos disponibles son and, or y not.

- Si termino temprano, y el tengo dinero, saldré de compras.
- Si tú estás en la pastelería o yo estoy en la pastelería, uno de nosotros comprará un pastel para papá.
- No estoy triste.

Operadores lógicos:

Los operadores lógicos se utilizan para realizar operaciones de lógica booleana entre expresiones y valores booleanos. Los operadores lógicos devuelven un valor booleano (True o False) según el resultado de la operación. Python cuenta con tres operadores lógicos principales: AND, OR y NOT.

Operador lógico AND (and):

El operador and devuelve True si ambas expresiones son verdaderas (True), de lo contrario, devuelve False. En otras palabras, la expresión con el operador and será verdadera solo si todas las condiciones son verdaderas.

```
a = 2
b = 16
c = 17

# Comprobando si a es menor que b y b es menor que c
resultado = a < b and b < c
# El resultado será True
# ya que ambas condiciones son verdaderas
# (2 es menor que 16 y 16 es menor que 17)
```

Operadores lógicos:

Los operadores lógicos se utilizan para realizar operaciones de lógica booleana entre expresiones y valores booleanos. Los operadores lógicos devuelven un valor booleano (True o False) según el resultado de la operación. Python cuenta con tres operadores lógicos principales: AND, OR y NOT.

Operador lógico OR (or):

El operador or devuelve True si al menos una de las expresiones es verdadera (True), si ambas expresiones son falsas (False), entonces devuelve False. En otras palabras, la expresión con el operador or será verdadera si al menos una de las condiciones es verdadera.

```
a = 2
b = 16
c = 17

# Comprobando si a es menor que b y b es menor que c
resultado = a < b or b > c
# El resultado será True
# ya que una de las condiciones es verdadera
# (2 es menor que 16 o 16 es mayor que 17)
```

Operadores lógicos:

Los operadores lógicos se utilizan para realizar operaciones de lógica booleana entre expresiones y valores booleanos. Los operadores lógicos devuelven un valor booleano (True o False) según el resultado de la operación. Python cuenta con tres operadores lógicos principales: AND, OR y NOT.

Operador lógico NOT (not):

El operador not es un operador unario que invierte el valor de una expresión booleana. Si la expresión es verdadera, el operador not devuelve False, y si la expresión es falsa, el operador not devuelve True.

```
casado = False

# Comprobando si no eres casado
resultado = not casado
# El resultado será True
# ya que estamos negando que eres casado
# (la variable casado es False)
```

A practicar...

Ejercicios:

De acuerdo con el contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

1. Solicite al usuario el ingreso de 2 números y retorne un mensaje en pantalla indicando cual es el mayor o que son iguales en el caso que así sea:

```
numero1 = int(input("Ingrese el primer número: "))
numero2 = int(input("Ingrese el segundo número: "))

if numero1 > numero2:
    print("El número mayor es",numero1)
elif numero1 < numero2:
    print("El número mayor es",numero2)
else:
    print("Los dos números son iguales.")
```

```
Ingrese el primer número: 16
Ingrese el segundo número: 32
El número mayor es 32
```

```
Ingrese el primer número: 16
Ingrese el segundo número: 8
El número mayor es 16
```

```
Ingrese el primer número: 11
Ingrese el segundo número: 11
Los dos números son iguales.
```

Ejercicios:

De acuerdo al contenido visto en clases, desarrolla scripts en lenguaje Python para los siguientes ejercicios:

2. Solicite al usuario el ingreso de un número y muestre por pantalla el cálculo de su Factorial paso a paso:

```python
numero = int(input("Ingrese un número entero positivo: "))

# Validar que el número ingresado sea positivo
if numero < 0:
    print("Solo se permiten números enteros positivos para calcular su factorial.")
else:
    # Inicializar el resultado como 1, ya que el factorial de 0 es 1
    resultado = 1

    # Calcular el factorial utilizando un bucle 'for'
    for i in range(1, numero + 1):
        # Imprimimos el paso a paso del proceso
        print(resultado,'X',i,'=',resultado * i)
        resultado *= i

    # Imprimir el resultado
    print(f"El factorial de {numero} es: {resultado}")
```

```
Ingrese un número entero positivo: 8
1 X 1 = 1
1 X 2 = 2
2 X 3 = 6
6 X 4 = 24
24 X 5 = 120
120 X 6 = 720
720 X 7 = 5040
5040 X 8 = 40320
El factorial de 8 es: 40320
```

inacap.cl