ASICI

Asociación Internacional de Ciberseguridad

Ejemplos de Python nivel Básico

1. Hola Mundo

```
print("Hola, Mundo!")
```

Este es el programa más básico en cualquier lenguaje de programación. Usa la función `print()` para mostrar el mensaje "Hola, Mundo!" en la consola.

2. Suma de dos números

```
a = 5
b = 3
print(a + b)
```

Este código asigna los valores 5 y 3 a las variables `a` y `b`, respectivamente, y luego utiliza la función `print()` para mostrar la suma de estos dos valores.

3. Entrada de usuario

```
nombre = input("¿Cuál es tu nombre? ")
print(f"Hola, {nombre}!")
```

Aquí se utiliza la función `input()` para obtener datos del usuario, que en este caso es su nombre. Luego, se imprime un saludo personalizado usando la variable `nombre` dentro de una cadena con formato (`f-string`).

4. Condicional `if`

```
x = 10
if x > 5:
    print("x es mayor que 5")
```

Este ejemplo demuestra una estructura de control básica. El código evalúa si `x` es mayor que 5, y si es así, imprime un mensaje. Si no se cumple la condición, el programa no hace nada.

5. Ciclo `for`

```
for i in range(5):
    print(i)
```

Este es un ciclo `for` que itera sobre una secuencia generada por `range(5)`, que produce los números del 0 al 4. En cada iteración, imprime el valor de `i`.

6. Ciclo `while`

```
i = 0
while i < 5:
    print(i)
    i += 1
```

El ciclo `while` continúa ejecutándose mientras la condición `i < 5` sea verdadera. En cada iteración, imprime el valor de `i` y lo incrementa en 1.

7. Función simple

```
def saludar(nombre):
    print(f"Hola, {nombre}")
saludar("María")
```

Este código define una función llamada `saludar` que toma un parámetro `nombre`. Cuando se llama a la función, imprime un saludo personalizado. En este caso, se llama a la función con el argumento "María".

8. Lista

```
frutas = ["manzana", "plátano", "cereza"]
print(frutas[0])
```

Aquí se crea una lista llamada `frutas` con tres elementos. Luego se accede al primer elemento de la lista usando el índice `[0]` y se imprime ("manzana").

9. Diccionario

```
persona = {"nombre": "Juan", "edad": 30}
print(persona["nombre"])
```

Este código crea un diccionario llamado `persona` con dos claves: `"nombre"` y `"edad"`. Después, imprime el valor asociado a la clave `"nombre"` ("Juan").

10. Operador ternario

```
edad = 18
mensaje = "Mayor de edad" si edad >= 18 else "Menor de edad"
print(mensaje)
```

Este ejemplo utiliza un operador ternario para asignar un valor a la variable `mensaje` dependiendo de si la condición `edad >= 18` es verdadera o falsa. Si es verdadera, el valor será `"Mayor de edad"`, si es falsa, `"Menor de edad"`.

11. Comprehensión de listas

```
cuadrados = [x**2 for x in range(5)]
print(cuadrados)
```

Aquí se usa una comprensión de lista para crear una lista de los cuadrados de los números del 0 al 4. Es una forma compacta y eficiente de generar listas.

12. Iterar sobre un diccionario

```
persona = {"nombre": "Ana", "edad": 25}
for clave, valor in persona.items():
    print(f"{clave}: {valor}")
```

Este código itera sobre los elementos de un diccionario utilizando el método `items()`, que devuelve pares clave-valor. Imprime cada clave y su valor asociado.

13. Función lambda

```
doble = lambda x: x * 2
print(doble(5))
```

Una lambda es una función anónima de una sola línea. En este caso, la función toma un número `x` y devuelve el doble de `x`. Luego se llama con el valor `5`.

14. Manejo de excepciones

```
try:
    resultado = 10 / 0
except ZeroDivisionError:
    print("No se puede dividir por cero")
```

Este ejemplo muestra cómo manejar errores en Python. Intenta dividir por cero dentro de un bloque `try`. Como eso genera una excepción `ZeroDivisionError`, se captura y se muestra un mensaje de error.

15. Abrir y leer un archivo

```
with open('archivo.txt', 'r') as archivo:
    contenido = archivo.read()
    print(contenido)
```

Este código abre un archivo en modo lectura (`'r'`), lee su contenido y lo imprime. El uso de `with` asegura que el archivo se cierre automáticamente después de usarlo.

16. Escribir en un archivo

```
with open('archivo.txt', 'w') as archivo:
    archivo.write("Hola desde Python")
```

Este ejemplo abre un archivo en modo escritura (`'w'`). Si el archivo no existe, lo crea; si ya existe, lo sobrescribe. Escribe la cadena "Hola desde Python" en el archivo.

17. Verificar si un número es par

```
numero = 4
if numero % 2 == 0:
    print("Es par")
else:
    print("Es impar")
```

Este código verifica si el número es par usando el operador de módulo (`%`). Si el residuo de la división por 2 es 0, el número es par; de lo contrario, es impar.

18. Generar números aleatorios

```
import random
print(random.randint(1, 10))
```

Aquí se usa la biblioteca `random` para generar un número aleatorio entero entre 1 y 10, y luego se imprime.

19. Ordenar una lista

```
numeros = [3, 1, 4, 1, 5]
numeros.sort()
print(numeros)
```

Este código ordena una lista de números de menor a mayor utilizando el método `sort()`.

20. Convertir texto a mayúsculas

```
texto = "hola mundo"
print(texto.upper())
```

Este código convierte una cadena de texto en minúsculas a mayúsculas utilizando el método `upper()`.

21. Calcular factorial

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
print(factorial(5))
```

Este es un ejemplo de una función recursiva que calcula el factorial de un número. La función se llama a sí misma hasta que `n` llega a 0, momento en que devuelve 1.

22. Generar una lista de números pares

```
pares = [x for x in range(20) if x % 2 == 0]
print(pares)
```

Este es otro ejemplo de comprensión de listas. Genera una lista de números pares entre 0 y 19.

23. Invertir una lista

```
lista = [1, 2, 3, 4, 5]
print(lista[::-1])
```

Este código invierte una lista usando el "slice" (`[::-1]`), que es una forma rápida de obtener la lista al revés.

24. Sumar los elementos de una lista

```
numeros = [1, 2, 3, 4, 5]
print(sum(numeros))
```

Aquí se usa la función `sum()` para sumar todos los elementos de una lista y luego imprimir el resultado.

25. Módulo de una función

```
def modulo(a, b):
    return a % b
print(modulo(10, 3))
```

Este código define una función que calcula el módulo de dos números (el residuo de la división). Luego se llama a la función con los valores `10` y `3`.

26. Generar una lista de números del 1 al 10

```
numeros = list(range(1, 11))
print(numeros)
```

Se usa `range()` para generar los números del 1 al 10, y `list()` convierte el rango en una lista.

27. Uso de `break` en un ciclo

```
for i in range(10):
    if i == 5:
        break
    print(i)
```

Este código utiliza `break` para salir del ciclo cuando `i` es igual a 5. Los números del 0 al 4 se imprimen, pero el ciclo termina cuando `i` llega a 5.

28. Uso de `continue` en un ciclo

```
for i in range(10):
    if i % 2 == 0:
        continue
    print(i)
```

El uso de `continue` salta a la siguiente iteración del ciclo si `i` es un número par. Por lo tanto, solo se imprimen los números impares.

29. Comprobar si un elemento está en una lista

```
frutas = ["manzana", "plátano", "cereza"]
if "manzana" in frutas:
    print("Sí, la manzana está en la lista")
```

Este código verifica si `"manzana"` está en la lista `frutas` usando el operador `in`. Si es así, imprime un mensaje.

30. Contar la frecuencia de elementos en una lista

```
from collections import Counter
lista = [1, 2, 2, 3, 3, 3]
frecuencias = Counter(lista)
print(frecuencias)
```

Se usa la clase `Counter` del módulo `collections` para contar la frecuencia de cada elemento en la lista. El resultado es un diccionario con los elementos como claves y las frecuencias como valores.