Jala University
Cursos de Admisión

ESTRUCTURAS DE CONTROL **CONDICIONALES Y REPETITIVAS**

Jala University
Cursos de Admisión

Parte I

Estructuras de control
Condicionales

A continuacion hablaremos de las estructuras de control condicionales

Que son las Estructuras de Control Condicionales?

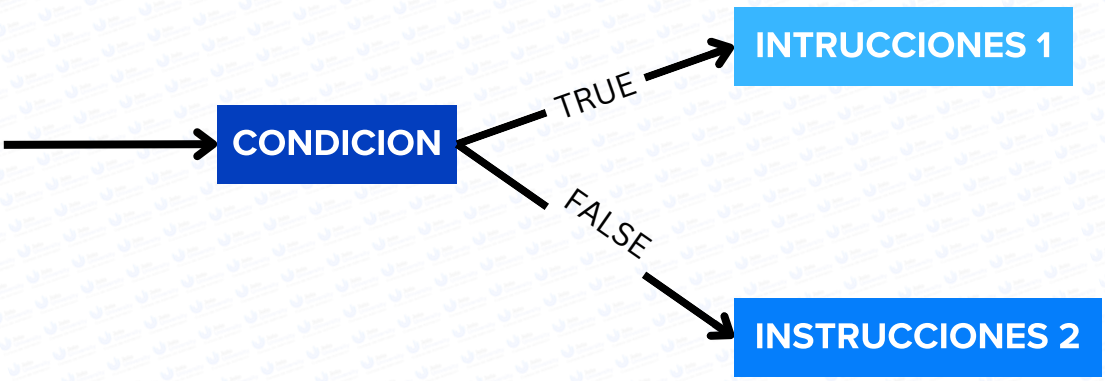
Estas estructuras modifican el flujo de ejecución del código, permitiendo que diferentes bloques de instrucciones se ejecuten dependiendo de si se cumplen ciertas condiciones o no. Para su uso en python usamos las palabras reservadas **if, else, elif**

Cual es la sintaxis de una condicional en python?

```python
esta_lloviendo = True

if esta_lloviendo:
    ## Ejecutar si la condicion es True
    print('Llevar paraguas')
else:
    ## Ejecutar si la condicion es False
    print('no hay avisos de lluvia')
```

Como se ejecuta el codigo?



Que tipos de condiciones se evaluan?

Se evaluan los operadores logicos y relacionales que dos dan como resultado **True** o **False**

El uso del elif

En Python, "elif" es una abreviatura de "else if" y se utiliza en las estructuras condicionales para evaluar múltiples condiciones en orden. Si la condición del "if" no es verdadera, se verifica la condición del "elif". Si la condición del "elif" es verdadera, se ejecuta el bloque de código asociado; de lo contrario, se puede pasar a otra cláusula "elif" o, si no hay más, al bloque "else". En resumen, "elif" permite manejar múltiples casos en una estructura condicional.

```
mes = 12

if mes == 1:
    print('es enero')
elif mes == 2:
    print('es febrero')
elif mes == 3:
    print('es marzo')
else:
    print('no sabemos en que mes estamos')
```

Expression Truthy y Falsy

Ya sea usando la condicional simple (if) o la multiple (elif) todas las condiciones como resultado deben ser **True** o **False**, es por ello que existen algunas evaluaciones que sin ser necesariamente una expresion condicional dan como resultado **True** o **False**, a estas se conocen como **Truthy** y **Falsy**

`True`	TRUE
`False`	FALSE
`1`	TRUE
`0`	FALSE
`-1`	TRUE
`"True"`	TRUE
`"False"`	TRUE
`1`	TRUE
`0`	TRUE
`-1`	TRUE
	FALSE
`None`	FALSE
`inf`	TRUE
`-inf`	TRUE
`[]`	FALSE
`{}`	FALSE
`[[]]`	TRUE
`[0]`	TRUE
`[1]`	TRUE

Jala University
Cursos de Admisión

Parte II

Estructuras de Control Repetitivas

Que son las Estructuras de Control Repetitivas?

Las estructuras de control repetitivas (también llamadas cíclicas o bucles), permiten ejecutar un mismo código, de forma repetida, cierta cantidad de veces mientras se cumpla una condición. Hablaremos de dos bucles: el **while** y el **for**

Bucle while

El bucle "**while**" se utiliza para repetir un bloque de código mientras una condición sea verdadera. Mientras la condición sea verdadera, el código dentro del bucle se ejecuta una y otra vez. Es importante asegurarse de que la condición cambie eventualmente para evitar un bucle infinito. En resumen, el bucle "**while**" permite la repetición controlada de un conjunto de instrucciones mientras se cumpla una condición específica.

Se usa bajo la siguiente sintaxis

```
numero = 2

while numero <= 10:
    print('el numero es:' + 2)
    numero += 1
```

Como ya mencionamos eventualmente la condición debe ser False, caso contrario se ingresa a un bucle infinito, otra forma de romper el ciclo es usando la palabra reservada **break**, te hará que el ciclo se rompa sin importar la condición

```
numero = 2

while numero <= 10:
    if numero == 3:
        break
    print('el numero es:' + 2)
    numero += 1
```

Bucle for

El bucle "**for**" se utiliza para iterar sobre una secuencia de elementos, como una lista o una cadena de texto. Permite ejecutar un bloque de código para cada elemento de la secuencia. En resumen, el bucle "**for**" simplifica la tarea de realizar operaciones repetitivas sobre los elementos de una colección.

La sintaxis es la siguiente

```
frutas = ['Manzana', 'Pera', 'Uva', 'Naranja']

for fruta in frutas:
    print('La fruta es: ' + fruta)
```

A diferencia del ciclo **while**, no necesitamos una condicion, ya que el ciclo terminara una vez se evalue el ultimo elemento de la lista, pero de igual forma que en el ciclo **while**, podemos romper el ciclo antes usando la palabra reservada **break**

```
frutas = ['Manzana', 'Pera', 'Uva', 'Naranja']

for fruta in frutas:

    if fruta == 'Pera':
        break

    print('La fruta es: ' + fruta)
```

Funcion range

range en Python es una función que crea una secuencia de números en un rango específico. Se utiliza comúnmente con bucles "for" para iterar sobre una secuencia de números. La función range toma hasta tres argumentos: el valor inicial, el valor final (no incluido en la secuencia), y el paso (intervalo entre los números). Puedes pensar en ello como una manera conveniente de generar una secuencia de números.
Por ejemplo:

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

Las "palabras reservadas" en programación son términos especiales que tienen un significado específico en el lenguaje de programación. No puedes usar estas palabras para nombrar variables u otras partes del código, ya que ya están reservadas para realizar funciones específicas en el programa. Son esenciales para que el lenguaje funcione correctamente y para establecer reglas coherentes en el código. Ejemplos comunes incluyen "if" (si), "else" (si no) y "for" (para).