ESTRUCTURAS DE CONTROL

Estructuras de Control

Condicionales

(toma de decisiones)

Iterativas

(repetición de acciones)

- Estructuras de Control Condicionales (i): *if*

Las estructuras de control condicionales sirven para tomar decisiones en función del resultado de la evaluación de la expresión o condición indicada, que será un valor booleano: *True* o *False*.

La estructura más básica de una expresión condicional es la siguiente:

```
if expresión:                  # la evaluación de la expresión dará como resultado True o False.
    sentencias_a_ejecutar      # si la evaluación es True entonces se ejecutará esta parte del código.
```

```
# Ejemplo 01

if x > 0:
   print("x es positivo")
```

- Si la expresión evaluada, *x > 0* es *True,* se ejecuta el *print*. Dicho de otro modo: si *x > 0* entonces se mostrará en pantalla el mensaje *"x es positivo".*
- Por el contrario, si la expresión *x > 0* es *False,* es decir, en caso de que el valor de x sea menor o igual que 0, entonces no se ejecuta el *print*.

- Estructuras de Control Condicionales (ii): *elif*

La sentencia *elif expresión* sirve para expresar nuevas condiciones en caso de que el *if* no se cumpla, es decir, el resultado de evaluar la expresión del *if expresión* sea *False.*

```
if expresión:                # la evaluación de la expresión dará como resultado True o False.
    sentencias_a_ejecutar    # si la evaluación es True entonces se ejecutará esta parte del código.
elif expresión:              # se evalúa el elif si la condición del if fue False.
    sentencias_a_ejecutar    # se ejecuta en caso de que la evaluación de la expresión de elif sea True.
```

```
# Ejemplo 02

if x > 0:
   print("x es positivo")
elif x < 0:
   print("x es negativo")
```

- Si la expresión evaluada, *x > 0* es *True,* se ejecuta la instrucción *print("x es positivo").*
- Por el contrario, si la expresión *x > 0* es *False,* se ejecuta la instrucción *elif x < 0.* Ahora, si *x* es negativo se imprimirá en pantalla *"x es negativo".*
- En caso de que x sea 0 no se realizará ninguna acción, ya que dicho caso no está contemplado en el Ejemplo.

- Estructuras de Control Condicionales (iii): múltiples *elif*

Podemos comprobar tantos casos como sea necesario encadenando sentencias *elif.*

```
# Ejemplo 03

valor = int(input('Introduce un número del 1 al 6: '))
if valor == 1:
    print('UNO')
elif valor == 2:
    print('DOS')
elif valor == 3:
    print('TRES')
elif valor == 4:
    print('CUATRO')
elif valor == 5:
    print('CINCO')
elif valor == 6:
    print('SEIS')
else:
    print('El numero introducido no esta entre 1 y 6')
```

- Estructuras de Control Condicionales (iv): *else*

La sentencia *else* se utiliza para indicar las acciones a realizar cuando las sentencias *if* y *elif* precedentes no han sido *True.*

```
if expresión:                 # la evaluación de la expresión dará como resultado True o False.
    sentencias_a_ejecutar     # si la evaluación es True entonces se ejecutará esta parte del código.
elif expresión:               # se evalúa el elif si la condición del if fue False.
    sentencias_a_ejecutar     # se ejecuta en caso de que la evaluación de la expresión de elif sea True.
else:                         # se ejecuta en caso de que todas las condiciones anteriores sean False.
    sentencias_a_ejecutar
```

```
# Ejemplo 04

if x > 0:
   print("x es positivo")
elif x < 0:
   print("x es negativo")
else:
   print("x es cero")
```

- El *else* se evalúa en caso de el *if* y los *elif* anteriores hayan sido *False.*

Nota: La cláusula *else* tiene que ir la última y no evalúa ninguna expresión.

- Estructuras de Control Condicionales (v): condicional anidado

Se pueden establecer niveles de anidación mezclando las diferentes construcciones condicionales.

```
# Ejemplo 05 v1

if x > 0:
    if y > 0:
        print("x e y son positivos")
    else:
        if y < 0:
            print("x e y tienen distinto signo")
else:
    if y < 0:
        print("x e y son negativos")
    else:
        print("x e y tienen distinto signo")
```

```
# Ejemplo 05 v2

if x > 0 and y > 0:
    print("x e y son positivos")
elif x < 0 and y < 0:
    print("x e y son negativos")
else:
    print("x e y tienen distinto signo")
```

Sin embargo, los condicionales anidados pueden resultar difíciles de leer y comprender rápidamente. Por tanto, conviene evitarlos utilizando construcciones más simples, por ejemplo, con la ayuda de los operadores lógicos.

- Estructuras de Control Iterativas (i)

Las estructuras de control iterativas (también denominadas bucles), permiten repetir bloques de código de forma cíclica.

while

for

- Estructuras de Control Iterativas (ii): bucle *while*

El bucle *while* realiza iteraciones mientras se cumpla una determinada condición lógica.

```
while expresión:          # la evaluación de la expresión dará como resultado True o False.
    sentencias            # si la evaluación es True entonces se ejecutará esta parte del código.
```

```
# Ejemplo 06

x = 1
while x < 6:
    print(x)
    x += 1
```

- Este bucle *while* se ejecutará mientras que el valor de la variable *x* sea inferior a 6.
- El bucle se ejecutará 5 veces e imprimirá en pantalla los valores: 1 2 3 4 5

- Estructuras de Control Iterativas (iii): bucle *while* infinito

A veces no podemos conocer o controlar a priori la condición de parada del bucle. En estos casos podemos utilizar un bucle *while* que se ejecute indefinidamente al establecer una condición *True*.

La parada se realizará en el interior del bucle mediante la instrucción *break.*

```
while True:
    sentencias
    if condición_parada: break          # el break rompe la ejecución del while
```

```
# Ejemplo 07

while True:
    n = int(input("Introduzca un numero (0 = fin):"))
    if n == 0: break
    x = 10 / n
    print(x)
```

- Este bucle *while* se ejecutará indefinidamente hasta que el usuario introduzca el valor 0, en cuyo caso se ejecutará la instrucción *break* y se saldrá del bucle a la siguiente instrucción del programa.

- Estructuras de Control Iterativas (iv): *continue*

Podemos saltar la ejecución de ciertas instrucciones del bucle a través de la sentencia *continue*, que nos lleva inmediatamente a la siguiente iteración del bucle.

```
# Ejemplo 08

""" Cuenta los numeros impares introducidos por teclado, ignorando los pares
    La ejecución termina cuando se introduce un 0
"""
n = None
impares = 0
while n != 0:
    n = int(input("Introduzca un numero (0 = fin):"))
    if n % 2 == 0: continue
    impares +=1
print("Números impares introducidos:", impares)
```

- Estructuras de Control Iterativas (v): bucle *for*

El bucle *for* realiza iteraciones sobre los elementos de cualquier secuencia (lista, cadena, diccionario, …).

```
for variable_iteración in secuencia_iterable:      # para cada elemento de la secuencia se ejecutarán las
    sentencias                                     # sentencias del bucle
```

La variable que va recorriendo la secuencia se denomina variable de iteración.

```
# Ejemplo 09

for i in [1, 5, 7, 9]:
    print(i)
```

- La variable de iteración *i* irá tomando los valores de los elementos de la secuencia especificada en el *for* y ejecutará la instrucción *print* correspondiente.
- El bucle realizará 4 iteraciones e imprimirá en pantalla los valores: 1 5 7 y 9.

- Estructuras de Control Iterativas (vi): *range()*

En Python 3, *range* sirve para generar una sucesión aritmética de números enteros.

range([inicio], fin, [paso]):

inicio: número inicial a partir del cual se generará la sucesión. El valor de *inicio* se incluye en la sucesión. El valor de *inicio* es *0* por defecto.
fin: número hasta el que ha de generarse la sucesión. El valor de *fin* no se incluye en la sucesión.
paso: salto entre el elemento *n* y el *n+1,* es decir, si vamos de 2 en 2, de 3 en 3, etc. El valor de *paso* es 1 por defecto

Para ver los valores del *range()*, es necesario convertirlo a lista con la función *list()*.

```
# Ejemplo 10: Ejemplos varios de uso de range

a=list(range(10))
b=list(range(1,11))
c=list(range(0,10,2))
d=list(range(1,10,2))
print(a)                # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(b)                # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(c)                # [0, 2, 4, 6, 8]
print(d)                # [1, 3, 5, 7, 9]
```

- Estructuras de Control Iterativas (vii): bucle *for* con *range()*

range es muy útil en la definición de bucles *for* ya que la secuencia numérica generada es iterable.

```
# Ejemplo 11

suma = 0
for i in (range(10)):
   suma += i
print("La suma de los 10 primeros numeros es:", suma)
```

- La variable de iteración *i* irá tomando los valores de los elementos de la secuencia generada por *range*.

```
# Ejemplo 12

for i in (range(-5,6)):
   print(i)
```

- Imprime en pantalla la secuencia numérica siguiente:
  -5 -4 -3 -2 -1 0 1 2 3 4 5

```
# Ejemplo 13: cuenta atrás para el lanzamiento

for i in (range(10,0,-1)):
   print(i)
```

- Imprime en pantalla la secuencia numérica siguiente:
  10 9 8 7 6 5 4 3 2 1

- Ejercicios.

1. Escribir un programa que permita al usuario introducir números hasta que el numero introducido sea -1. Para cada numero introducido indicar su calificación:

   [0 - 5) Suspenso
   [5 - 7) Aprobado
   [7 - 8.5) Notable
   [8.5 - 10] Sobresaliente

2. Escribir un programa para jugar a adivinar un número entero entre 1 y 100 almacenado en una variable. El jugador irá introduciendo números hasta adivinarlo. Para cada intento, el programa responderá si el número a adivinar es mayor o menor que el introducido.

3. Generar con *range()* las siguientes sucesiones:

   1,6,11,16,…,101
   -100,-90,-80,…,100
   25,23,21,…1

4. Escribir un programa que imprima en pantalla las 10 primeras potencias de 2.

5. Escribir un programa que imprima en pantalla el cuadrado de los números impares del 1 hasta el 20.

ESTRUCTURAS DE CONTROL

1. Escribir un programa que permita al usuario introducir números hasta que el numero introducido sea -1. Para cada numero introducido indicar su calificación:

   [0 - 5) Suspenso
   [5 - 7) Aprobado
   [7 - 8.5) Notable
   [8.5 - 10] Sobresaliente

```
# Ejercicio 1

num = None
while(True):
    num = float(input("Introduce nota:"))
    if(num == -1):
        print("Finalizando Ejecución")
        break
    if(num >= 0.0 and num < 5):
        print("Suspenso")
    elif(num >= 5 and num < 7):
        print("Aprobado")
    elif(num >= 7 and num < 8.5):
        print("Notable")
    elif(num >= 8.5 and num <= 10):
        print("Sobresaliente")
    else:
        print("Nota incorrecta")
```

2. Escribir un programa para jugar a adivinar un número entero entre 1 y 100 almacenado en una variable. El jugador irá introduciendo números hasta adivinarlo. Para cada intento, el programa responderá si el número a adivinar es mayor o menor que el introducido.

```
# Ejercicio 2

num1 = 34
while(True):
    num2 = int(input("Numero [1-100]:"))
    if(num2 < 1 or num2 > 100):
        print("Numero incorrecto")
        continue
    if(num2 == num1):
        print("NUMERO ADIVINADO")
        break
    elif(num2 > num1):
        print("El número buscado es menor")
    else:
        print("El número buscado es mayor")
```

3. Generar con *range()* las siguientes sucesiones:

1,6,11,16,…,101
-100,-90,-80,…,100
25,23,21,…1

```
# Ejercicio 3

range(1,102,5)

range(-100,101,10)

range(25,0,-2)
```

4. Escribir un programa que imprima en pantalla las 10 primeras potencias de 2.

```
# Ejercicio 4

for i in range(10):
   print("2 elevado a", i, "=", 2**i)
```

5. Escribir un programa que imprima en pantalla el cuadrado de los números impares del 1 hasta el 20.

```
# Ejercicio 5

for i in range(1,20,2):
   print(i, "al cuadrado =", i * i)
```