If... Elif... Else

Python permite control del flujo de la ejecución de un programa utilizando la instrucción condicional "if". Esta instrucción crea un bloque condicional.

In [ ]:

```
num = int(input("Ingrese un numero entero: "))

if num % 2 == 0:
    print("El numero ingresado es par")
else:
    print("El numero ingerado no es par")
```

Es importante notar que siempre se tiene la misma sintaxis:

```
if expresion:
    bloque de instrucciones
```

Luego de la expresión van ":" que indica que lo que sigue es un bloque que se encontrará sangrada a la derecha (segun el estandar, 4 espacios). Si una linea no esta sangrada, se considerará fuera del bloque

```
if expresion:
    bloque de instrucciones
else:
    bloque de instrucciones
```

En el caso que no se cumpla la expresión, se pude incluír una clausula "else" para considerar el caso contrario.

In [ ]:

```
num = int(input("Ingrese un numero entero: "))

if num < 50:
    print("El numero ingresado es menor que 50")
elif num < 100:
    print("El numero ingresado es menor que 100")
elif num < 200:
    print("El numero ingresado es menor que 200")
else:
    print("El numero ingresado es mayor que 200")
```

La instrucción "if" no tiene una marca de fin: el final de la instrucción esta dada por la ausencia de sangría, por lo que es importante mantener un orden de escritura

In [ ]:

```
edad = int(input("Ingrese su edad: "))
edadValida = False

if edad < 0:
    print("Edad ingresada invalida")
elif edad > 100:
    print("Edad ingresada invalida")
else:
    edadValida = True

if edadValida:
    if edad < 18:
        print("Ud. es menor de edad")
    else:
        print("Ud. es mayor de edad")
```

Como en cualquier lenguaje de programación se pueden incluír operadores de relación, así como instrucciones if anidadas:

In [ ]:

```
edad = int(input("Ingrese su edad: "))

if edad > 0:
    if edad < 100:
        if edad < 18:
            print("Ud. es menor de edad")
        else:
            print("Ud. es mayor de edad")
    else:
        print("Edad invalida")
else:
    print("Edad invalida")
```

In [ ]:

```
edad = int(input("Ingrese su edad: "))

if edad < 0 or edad > 100:
    print("Edad invalida")
else:
    if edad > 18 and edad < 70:
        print("Ud. es mayor de edad y puede conducir")
    elif edad >= 70:
        print("Ud. es mayor de edad y no puede conducir")
```

Todos los casos anteriores son una muestra que se puede construír cualquier lógica de control con la intruccion if...elif...else y las operaciones logicas y de relacion.

Python *if* y la prueba de membresía *in*

Una instrucción asociada a *if* es la instrucción *in* que permite verificar si un valor forma parte de un listado de valores:

In [ ]:

```
primos = (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 9
7)

num = int(input("Ingrese un numero entero [0 - 100]: "))
if num in primos:
    print("El numero ingresado es primo")
else:
    print("El numero ingresado no es primo")
```

La variable *primos* es una tupla. Esta es una secuencia de valores asociados a una sola variable. En otro documento se verá con detalle el tipo tupla en Python; lo que nos importa para este ejemplo es que el operador *in* retorna un valor logico verdadero si un valor es miembro de un grupo de valores (en este caso, una tupla). Esta operacion también funciona con listas

In [ ]:

```
primos = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 9
7]

num = int(input("Ingrese un numero entero [0 - 100]: "))
if num in primos:
    print("El numero ingresado es primo")
else:
    print("El numero ingresado no es primo")
```

Una lista es equivalente a una tupla, solo que los valores de una lista pueden modificarse (una tupla no acepta cambios en los valores).

El uso del operador *in* evita tener que buscar un valor a lo largo de una lista de valores, recorriendo cada valor y haciendo una comparación uno-a-uno como sucede en otros lennguajes de programación. El uso del operador *in* es un ejemplo de un codigo *pythonico*, es decir, una construcción elegante y legible que es preferida como código Python.

In [4]:

```
# Hay otra instruccion que puede parecer inutil

numero=0

if numero == 0:
    pass

# Esto no hara absolutamente nada. La instruccion pass es simplemente un "continuar". Sin embargo,
# el compilador no ignora esta sentencia como si lo hace con los comentarios. la palabra pass sirve par
a marcar algo
#que se quiere hacer posteriormente en el codigo.
```

In [5]:

```
#Ahora que podemos constuir codigos mas largos y tomando en cuenta lo importante que es el formato en P
ython.
#Existe una guia de estilo de escritura de codigo llamada PEP-8. Aqui algunos ejemplos de las buenas pr
acticas
#del PEP-8 (https://www.python.org/dev/peps/pep-0008/):

# - Usar cuatro espacios en lugar de indentación
# - Limitar el tamaño de las lineas de codigo a 79 caracteres
# - Separar con un espacio los argumentos, luego de la coma
# - Rodear los operadores con espacios en blanco (a excepcion de la definicion de opciones por defecto
 de los argumentos
#   de una funcion
```

In [ ]: