Módulo 3

Ejecución condicional, bucles, listas y su procesamiento con operadores lógicos.

En esta sección aprenderemos:

- Tomar decisiones en Python (if, if-else, if-elif, else).
- Repetir la ejecución de código usando los bucles (while, for).
- Cómo realizar operaciones lógicas y de bit a bit en Python.
- Listas en Python (construcción, indexación y segmentación; manipulación de contenido).
- Listas multidimensionales y sus aplicaciones.

evedo Cruz

Ejecución condicional

If, elif, else

Un mecanismo que nos permite hacer algo si se **cumple** una **condición**, y no hacerlo si no se cumple.

Con estas palabras, podemos **incluir** y controlar **varias condiciones**.

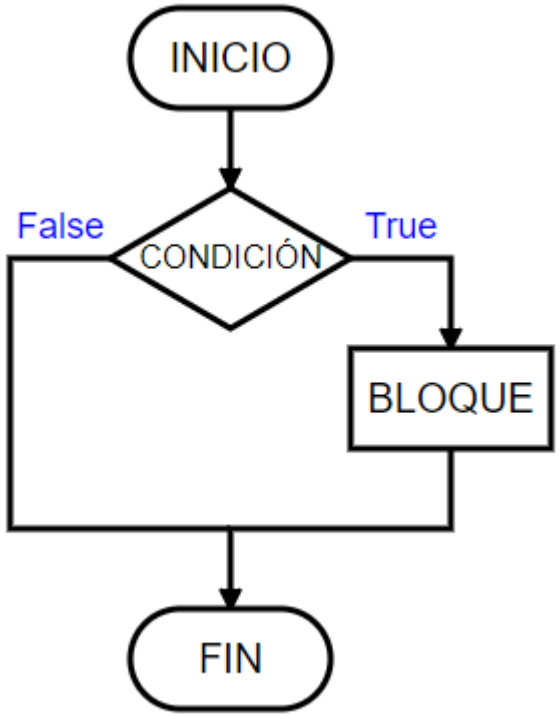
Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Condicional: IF

La estructura de control **if** ... permite que un programa ejecute unas instrucciones cuando se cumplan una condición.

En inglés "**if**" significa "si" (condición).



Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Condicional: IF

```
if verdadero_o_falso:
    hazlo_si_es_verdadero
```

```
if el_clima_es_bueno:
    salir_de_paseo()
comer_algo()
```

Recuerda que las **sentencias condicionales deben tener sangría**.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

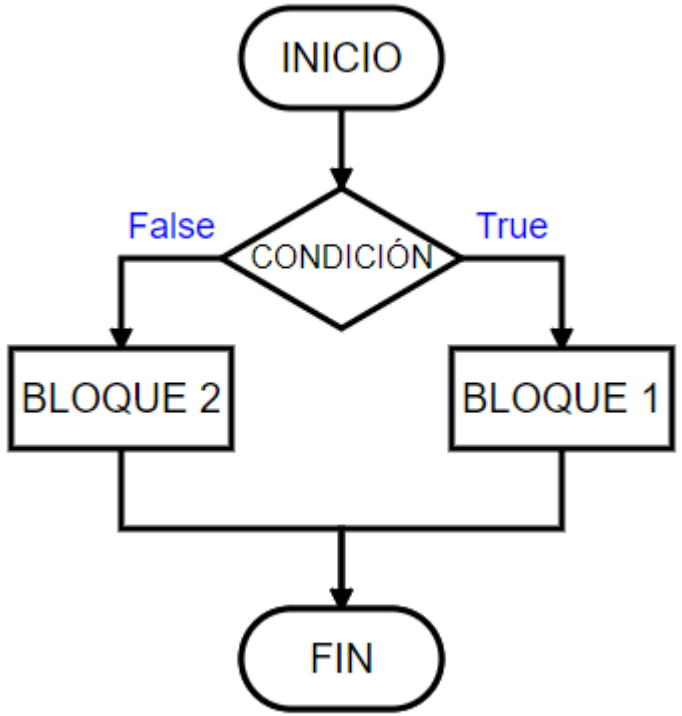
Condicional: IF

```
numero = int(input("Escriba un número positivo: "))

if numero < 0:
    print("¡Le he dicho que escriba un número positivo!")

print(f "Ha escrito el número {numero}")
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ejecución condicional: la sentencia if-else



La estructura de control **if ... else ...** permite que un programa ejecute una instrucción cuando se **cumple** una condición y otras instrucciones cuando **no se cumple** esa condición.

En inglés **if** significa "si" (condición)
y "**else**" significa "si no".

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ejecución condicional: la sentencia if-else

```
if verdadero_o_falso:
    hazlo_si_es_verdadero
else:
    hazlo_esto_si_es_falso
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ejecución condicional: la sentencia if-else

```
edad = int(input("¿Cuántos años tiene? "))
if edad < 18:
    print("Es usted menor de edad")
else:
    print("Es usted mayor de edad")
print("¡Hasta la próxima!")
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Bifurcación: la sentencia if-elif-else

elif:

elif se usa para verificar más de una condición, y para detener cuando se encuentra la primera sentencia verdadera.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Bifurcación: la sentencia if-elif-else

else:

No debes **usar** else **sin un** if precedente.

else siempre es la **última rama de la cascada**, independientemente de si has usado elif o no.

else es una parte **opcional** de la cascada, y puede omitirse.

Si hay una rama **else** en la cascada, solo se ejecuta una de todas las ramas.

Si no hay una rama else, es posible que no se ejecute ninguna de las opciones disponibles.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Bifurcación: la sentencia if-elif-else

```
if condición_1:
    bloque 1
elif condición_2:
    bloque 2
else:
    bloque 3
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

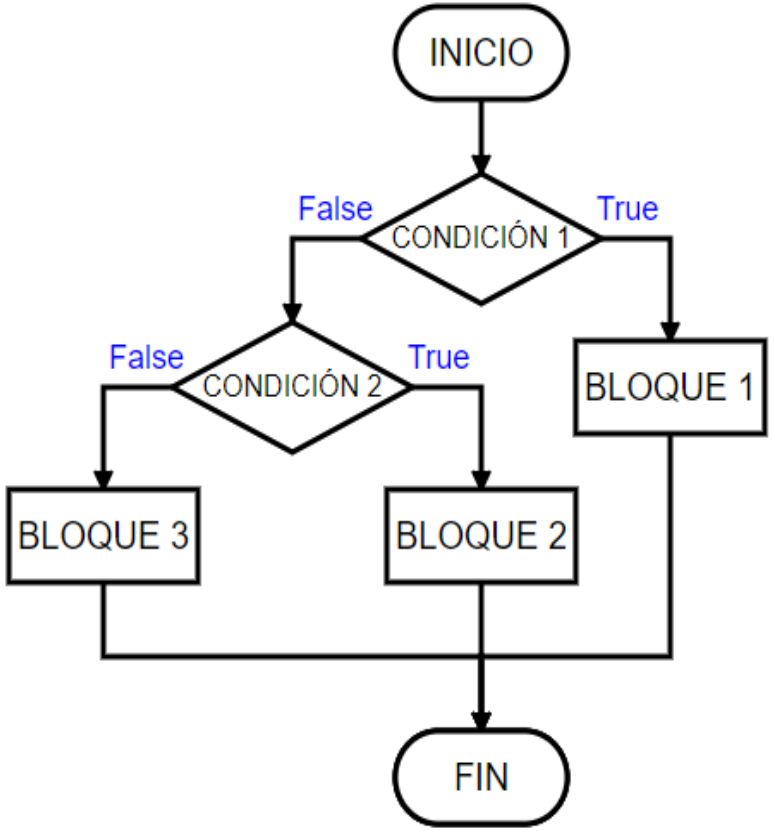
Bifurcación: la sentencia if-elif-else

```
if … elif … else

edad = int(input("¿Cuántos años tiene? "))
if edad >= 18:
    print("Es usted mayor de edad")
elif edad < 0:
    print("No se puede tener una edad negativa")
else:
    print("Es usted menor de edad")
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Sentencias if-else anidadas



Caso particular: la instrucción colocada después del **if** es otro **if**.

Este uso de la sentencia **if** se conoce como anidamiento;

Recuerda que cada **else** se refiere al **if** que se encuentra en el mismo nivel de sangría.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Sentencias if-else anidadas

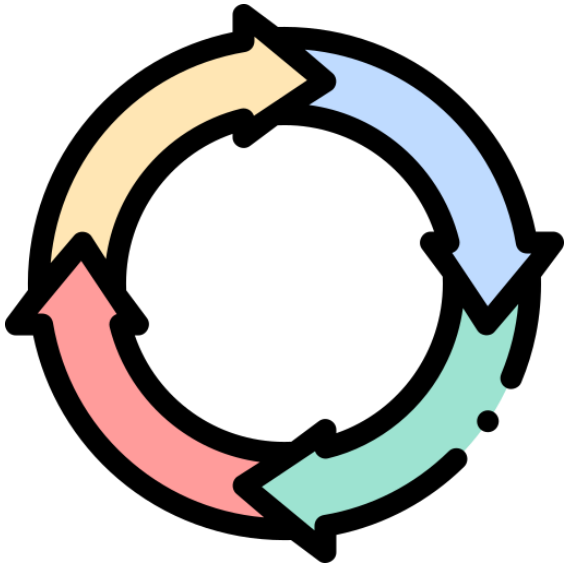
```
if el_clima_es_bueno:
    if ves_un_restaurante:
        comer_ahi()
    else:
        comprar_torta()
else:
    if si_hay_boletos:
        entrar_al_cine()
    else:
        ir_shopping()
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Introducción a los bucles (ciclos)

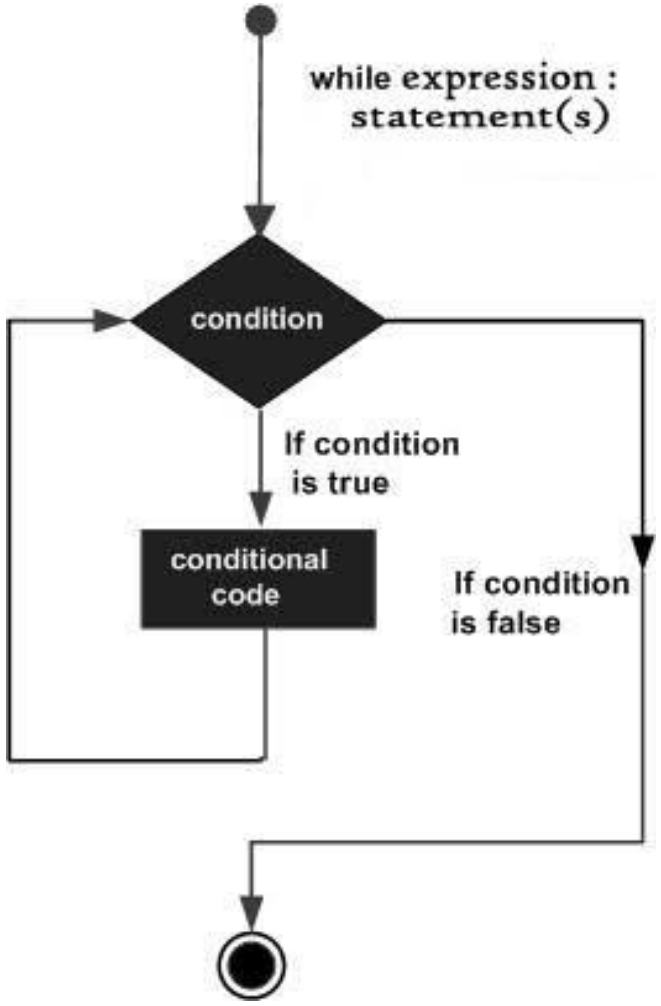
La ejecución de una determinada parte del código **más de una vez** se denomina **bucle**. El significado de este término es probablemente obvio para ti.



Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Estructuras Iterativas: Ciclo While



Estructura de **control** que permite **repetir** un bloque de código mientras se cumpla una condición específica.

La condición se **evalúa** al **principio** de cada iteración.

Si la condición es verdadera, el bloque de código se ejecuta; de lo contrario, se sale del bucle y continúa con la ejecución del programa.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Estructuras Iterativas: Ciclo While

```
contador = 1
while contador < 6:
    print(contador)
    contador = contador + 1
print("Bucle terminado")
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

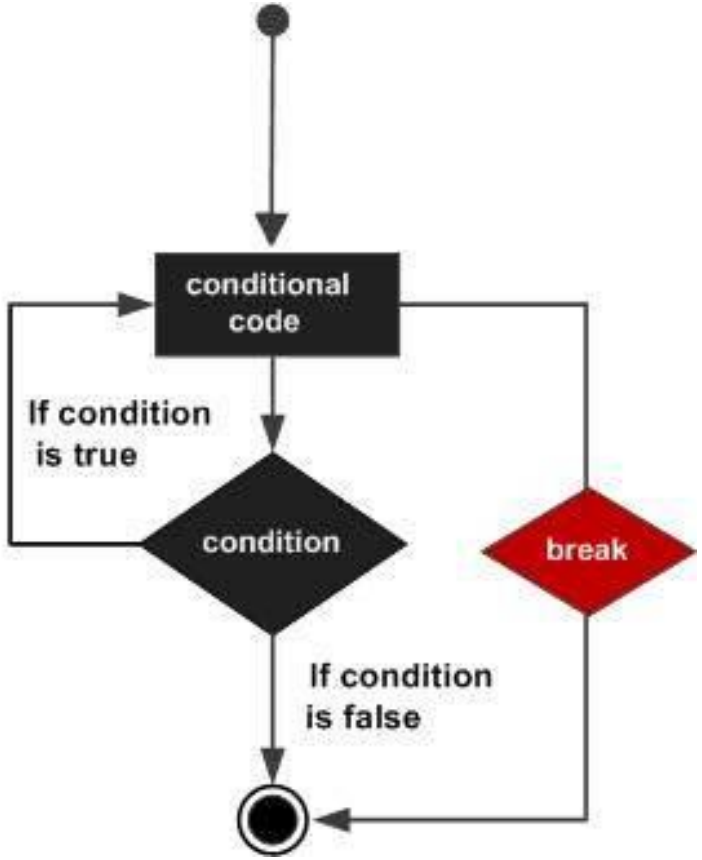
Un bucle infinito

```
while True:
    print("Estoy atrapado dentro de un bucle.")
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ciclo While: Break



Una forma de **controlar** la ejecución de un bucle en Python.

Cuando se encuentra la instrucción **break** dentro de un bucle, se sale inmediatamente del bucle, incluso si la condición del bucle aún se evalúa como verdadera.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ciclo While: Break

```
contador = 1
while contador < 6:
    if contador == 3:
        break
    print(contador)
    contador = contador + 1
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ciclo For

El bucle **for** se utiliza para recorrer los elementos de un objeto iterable (lista, tupla, conjunto, diccionario, ...) y ejecutar un bloque de código.

En cada paso de la **iteración** se tiene en cuenta a un **único** elemento del objeto iterable, sobre el cuál se pueden aplicar una serie de operaciones.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ciclo For

Sintáxis:

```
for <elem> in <iterable>:
    <Tu código>
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Iterables

Un **iterable** es un objeto que permite iterar sobre él, es decir, que se puede *recorrer sus elementos uno a uno*.

En Python, los tipos principales list, tuple, dict, set o string entre otros, son iterables, por lo que podrán ser usados en el bucle for.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Listas

Un tipo de dato que permite **almacenar** datos de cualquier tipo.

Son **mutables** y dinámicas, lo cual es la principal diferencia con los sets y las tuplas.

Una lista sea crea con **[ ]** separando sus elementos con comas .

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Range

La función **range**() genera una secuencia de números. Acepta enteros y devuelve objetos de rango.

La sintaxis de range() tiene el siguiente aspecto:

```
range(star,stop,step):
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Ciclo For in range

Sintáxis:

```
for n in range(1,8):
    print ("Hola ", n)
```

```
Hola  1
Hola  2
Hola  3
Hola  4
Hola  5
Hola  6
Hola  7
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Estructuras Iterativas, Ciclo For

Uso y sintaxis

Ejemplo en un range:

```
for n in range(1,8):
    print ("Hola ", n)
```

```
Hola  1
Hola  2
Hola  3
Hola  4
Hola  5
Hola  6
Hola  7
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Estructuras Iterativas, Ciclo For

Uso y sintaxis

Ejemplo en una lista:

```
nums = [4, 78, 9, 84]
for n in nums:
        print(n)
```

```
4
78
9
84
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Estructuras Iterativas, Ciclo For

Uso y sintaxis

Ejemplo en un diccionario (llave) :

```
alumnos = {"SOFY":19, "BOB":27, "ALEX":23}
for n in alumnos:
    print (n)
```

```
SOFY
BOB
ALEX
```

Ejemplo en un diccionario (valores) :

```
alumnos = {"SOFY":19, "BOB":27, "ALEX":23}
for n in alumnos.values():
    print (n)
```

```
19
27
23
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Estructuras Iterativas, Ciclo While

Ejemplo encontrar en una lista

```
valores = [5, 1, 9, 2, 7, 4]
encontrado = False
indice = 0
longitud = len(valores)
while not encontrado and indice < longitud:
    valor = valores[indice]
    if valor == 2:
        encontrado = True
    else:
        indice += 1
if encontrado:
    print(f'El número 2 ha sido encontrado en el índice {indice}')
else:
    print('El número 2 no se encuentra en la lista de valores')
```

Puntos clave:

- Lista
- Operadores lógicos
- Formateando la sentencia print

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Estructuras Iterativas, Ciclo While

Ejemplo encontrar en una lista

```
valores = [5, 1, 9, 2, 7, 4]
indice = 0
longitud = len(valores)
while indice < longitud:
    valor = valores[indice]
    if valor == 2:
        print(f'El elemento 2 ha sido encontrado en el índice {indice}')
        break
    else:
        indice += 1
else:
    print('El elemento 2 no se encuentra en la lista de valores')
```

Inténtalo tu mismo:

- Elimina el condicional encontrado y sustitúyelo por el operador break

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

ESTRUCTURAS DE CONTROL (CONDICIONANTES) IF, ELSE

EJERCICIO EN CLASE (1)

Requerimientos:

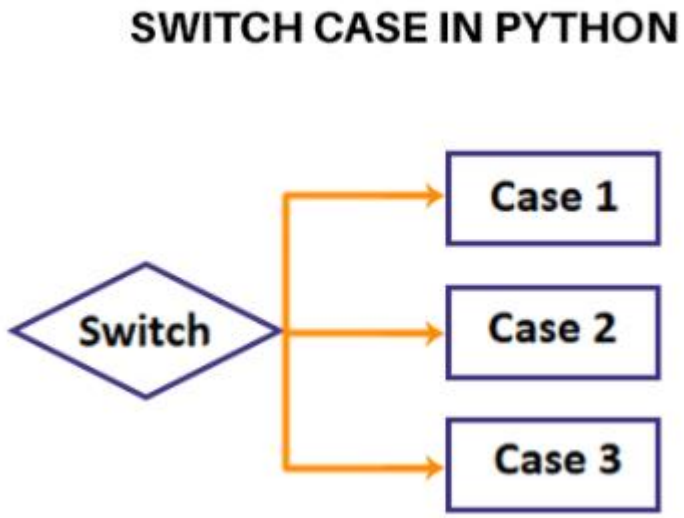
- Preguntar al usuario su nombre
- Darle la bienvenida
- Preguntar que operación le gustaría realizar (suma, resta, multiplicación, división)
- Solicitar valores para la operación y ejecutar la operación
- Regresar el resultado
- Despedirse del usuario

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

ESTRUCTURAS DE CONTROL **switch-case**

La sentencia **switch-case** es una característica poderosa que permite el control del flujo del programa basado en el valor de una variable o expresión. Se puede utilizar para ejecutar diferentes bloques de código, dependiendo del valor de la variable en tiempo de ejecución



Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

EJERCICIO EN CLASE (2)

Requerimientos:

Desarrollar una calculadora avanzada que permita realizar las siguientes operaciones:

- Suma, Resta, Multiplicación, División
- Cálculo del área de un circulo, aproximada y exacta mediante Pi
- Cálculo del área de un triangulo.
- Cálculo del perímetro de un cuadrado.

Características

- Uso de funciones para realizar las operaciones matemáticas
- Import de su módulo de funciones para los cálculos.
- Manejo de Errores
- Uso de comentarios y notación para el nombrado de variables

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

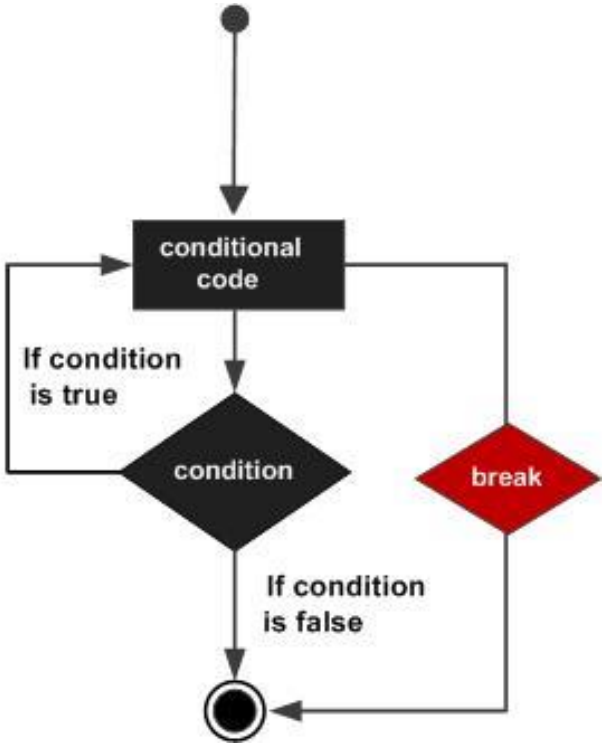
Estructuras Iterativas, Ciclo for..break

Uso y sintaxis

Ejemplo recorre hasta encontrar una coincidencia:

```
for n in range(1,10):
    if n == 6 :
       break
    else:
        print(n)

print ("fin del programa")
```

```
1
2
3
4
5
fin de programa
```



Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

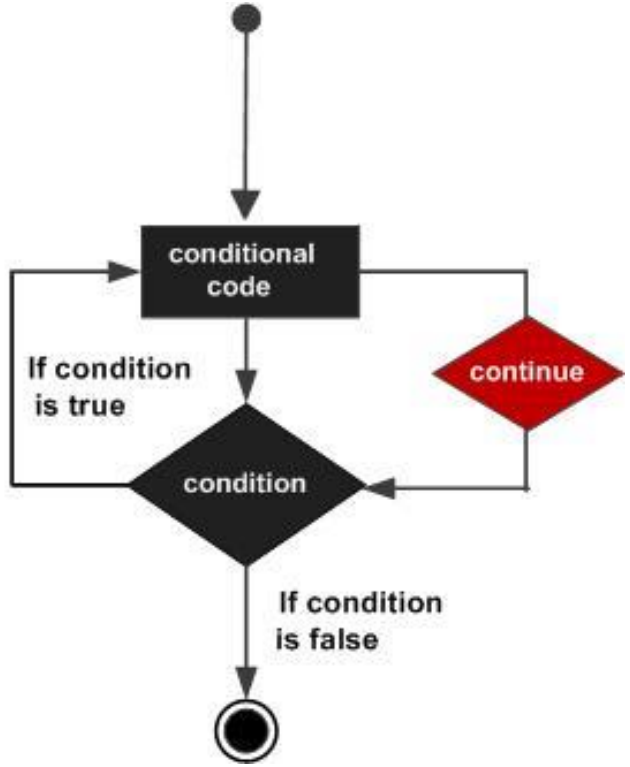
Estructuras Iterativas, Ciclo for..continue

Uso y sintaxis

Ejemplo recorre hasta encontrar una coincidencia:

```
for n in range(1,10):
    if n == 6 :
       continue
    else:
        print(n)

print ("fin del programa")
```

```
1
2
3
4
5
7
8
9
fin del programa
```

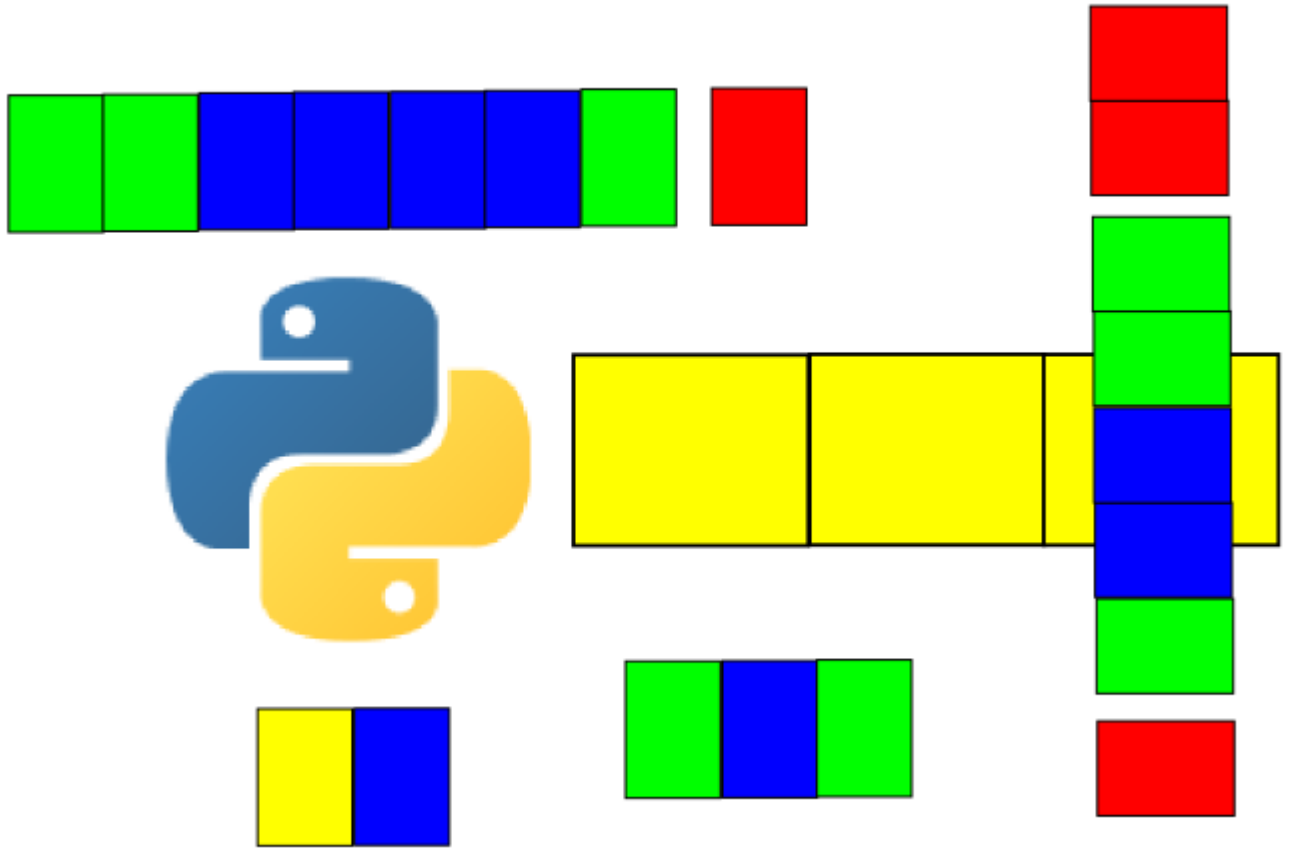


Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

¿Qué son las listas?

Las **listas** en **Python** son un tipo contenedor, compuesto, que se usan para almacenar conjuntos de elementos relacionados del mismo tipo o de tipos distintos. Junto a las clases tuple, range y str, son uno de los tipos de secuencia en **Python**, con la particularidad de que son mutables.



Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Laboratorio

Objetivo:

- Utilizar el bucle while.
- Reflejar situaciones de la vida real en código de computadora.

Escenario:

- Un mago junior ha elegido un número secreto. Lo ha escondido en una variable llamada secret_number. Quiere que todos los que ejecutan su programa jueguen el juego Adivina el número secreto, y adivina qué número ha elegido para ellos. ¡Quiénes no adivinen el número quedarán atrapados en un bucle sin fin para siempre! Desafortunadamente, él no sabe cómo completar el código.
- Tu tarea es ayudar al mago a completar el código en el editor de tal manera que el código:
- Pedirá al usuario que ingrese un número entero.
- Utilizará un bucle while.
- Comprobará si el número ingresado por el usuario es el mismo que el número escogido por el mago. Si el número elegido por el usuario es diferente al número secreto del mago, el usuario debería ver el mensaje "¡Ja, ja! ¡Estás atrapado en mi bucle!" y se le solicitará que ingrese un número nuevamente. Si el número ingresado por el usuario coincide con el número escogido por el mago, el número debe imprimirse en la pantalla, y el mago debe decir las siguientes palabras: "¡Bien hecho, muggle! Eres libre ahora".
- ¡El mago está contando contigo! No lo decepciones.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

Sintaxis

```
factura = ['pan', 'huevos', 100, 1234]
```

Nombre de la lista

Elementos mutables y heterogéneos

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

Sintaxis

```
factura          =          ['pan', 'huevos', 100, 1234]
```

Nombre de la lista

Elementos mutables y heterogéneos

```
factura = ['pan', 'huevos', 100, 1234]
print(factura)
['pan', 'huevos', 100, 1234]
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

¿Cómo acceder a sus elementos?

- Una lista en Python es una estructura de datos formada por una secuencia ordenada de objetos.
- Los elementos de una lista pueden accederse mediante su índice, siendo 0 el índice del primer elemento.

```
factura = ['pan', 'huevos', 100, 1234]

print(factura)
print(factura[0])
print(factura[3])
```

```
pan
1234
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

Ejemplo

Obtener el promedio y la desviación estándar de una serie numérica, contenida en la siguiente lista

```
edadesGrupo = [19,23,34,23,21,22,35,18]
```

Inténtalo tu mismo:

- Utiliza estructuras if ó while

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

Ejemplo

Obtener el promedio y la desviación estándar de una serie numérica, contenida en la siguiente lista

```
edadesGrupo = [19,23,34,23,”21”,22,35,18]
```

Inténtalo tu mismo:

- Utiliza estructuras if ó while
- En el caso de los valores inválidos, podemos no considerarlos para las operaciones

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

Ejemplo

Obtener el promedio y la desviación estándar de una serie numérica, contenida en la siguiente lista

```
edadesGrupo = [19,23,34,23,21,22,35,18]
```

```
import statistics
edadesGrupo = [19,23,34,23,21,22,35,18]
stD = statistics.stdev(edadesGrupo)
promedio = statistics.mean(edadesGrupo)
print (stD)
print (promedio)
```

```
6.5013734812620285
24.375
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

Métodos

```
edadesGrupo = [19,23,34,23,21,22,35,18]
#append
edadesGrupo.append(88)
print(edadesGrupo)
#sort
edadesGrupo.sort()
print(edadesGrupo)
#count
print(edadesGrupo.count(23))
#extend
edadesGrupo.extend(range(3))
print(edadesGrupo)
#index
print(edadesGrupo.index(88))
#insert
edadesGrupo.insert(3,"Selene")
print(edadesGrupo)
#pop
print(edadesGrupo.pop())
print(edadesGrupo)
#remove
edadesGrupo.remove(34)
print(edadesGrupo)
#reverse
edadesGrupo.reverse()
print(edadesGrupo)
```

Salida en terminal

```
[19, 23, 34, 23, 21, 22, 35, 18, 88]
[18, 19, 21, 22, 23, 23, 34, 35, 88]
2
[18, 19, 21, 22, 23, 23, 34, 35, 88, 0, 1, 2]
8
[18, 19, 21, 'Selene', 22, 23, 23, 34, 35, 88, 0, 1, 2]
2
[18, 19, 21, 'Selene', 22, 23, 23, 34, 35, 88, 0, 1]
[18, 19, 21, 'Selene', 22, 23, 23, 35, 88, 0, 1]
[1, 0, 88, 35, 23, 23, 22, 'Selene', 21, 19, 18]
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

Tipos de datos compuestos, Listas

Métodos

- append()
  - agrega un elemento al final de una lista.
- count()
  - recibe un elemento como argumento, y cuenta la cantidad de veces que aparece en la lista.
- extend()
  - extiende una lista agregando un iterable al final
- index()
  - recibe un elemento como argumento, y devuelve el índice de su primera aparición en la lista.
- insert()
  - inserta el elemento x en la lista, en el índice i.
- pop()
  - devuelve el último elemento de la lista, y lo borra de la misma.
- remove()
  - recibe como argumento un elemento, y borra su primera aparición en la lista.
- reverse()
  - reverse()
- sort()
  - ordena los elementos de una lista.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

EJERCICIO 1

En una determinada empresa, sus empleados son evaluados al final de cada año. Los puntos que pueden obtener en la evaluación comienzan en 0.0 y pueden ir aumentando, traduciéndose en mejores beneficios. Los puntos que pueden conseguir los empleados pueden ser **0.0, 0.4, 0.6 o más**, pero **no valores intermedios** entre las cifras mencionadas. A continuación se muestra una tabla con los niveles correspondientes a cada puntuación. La cantidad de dinero conseguida en cada nivel **es de 2.400 multiplicado por la puntuación del nivel**.

Nivel	Puntuación
Inaceptable	0.0
Aceptable	0.4
Meritorio	0.6 o más

Escribir un programa que lea la puntuación del usuario e indique su nivel de rendimiento, así como la cantidad de dinero que recibirá el usuario.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

SOLUCIÓN

```
bonificacion = 2400
inaceptable = 0
aceptable = 0.4
meritorio = 0.6
puntos = float(input("Introduce tu puntuación: "))
# Clasifiación por niveles de rendimiento
if puntos == inaceptable:
   nivel = "Inaceptable"
elif puntos == aceptable:
   nivel = "Aceptable"
elif puntos >= 0.6:
   nivel = "Meritorio"
else:
   nivel = ""
# Mostrar nivel de rendimiento
if nivel == "":
   print("Esta puntuación no es válida")
else:
   print("Tu nivel de rendimiento es = " , nivel)
   print("Te corresponde cobrar = " , (puntos * bonificacion))
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

RETOS DE TIEMPO

1. Crea un ciclo for que cuente de 0 a 100 de 3 en 3.
2. Imprime la tabla de multiplicar del 9 utilizando el ciclo for.
3. Imprime todas las tablas de multiplicar.
4. Crea un ciclo que imprima solo los números primos de 1 al 100.

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

EJERCICIOS DE LISTAS

- Escribir un programa que almacene las asignaturas de un curso (Matemáticas, Física, Química, Historia y Lengua) en una lista.

- Escribir un programa que almacene en una lista 5 carreras diferentes(Informática, Psicología, Comercio, Administración y Economía) , solicitar al usuario 5 nombres e imprimir en pantalla de la siguiente manera:
  - Sara estudia Informática
  - Lola estudia Psicología
  - Mario estudia Comercio
  - Jesús estudia Administración
  - María estudia Economía

- Escribir un programa que solicite al usuario 5 nombres , guárdalos en una lista e imprimirlos en pantalla

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

RETO DE TIEMPO

```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2
del list_1[0]
del list_2[0]

print(list_3)
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com

- RETO DE TIEMPO

```
list_1 = ["A", "B", "C"]
list_2 = list_1
list_3 = list_2
del list_1[0]
del list_2[:]

print(list_3)
```

Elaborado por: Mto. Humberto Acevedo Cruz

hac106@hotmail.com