PYTHON

Instructivo # 4

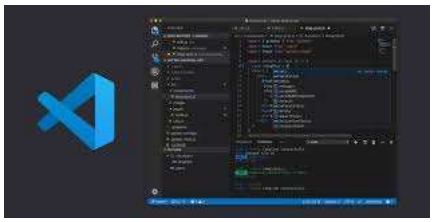
Ciclos en Python

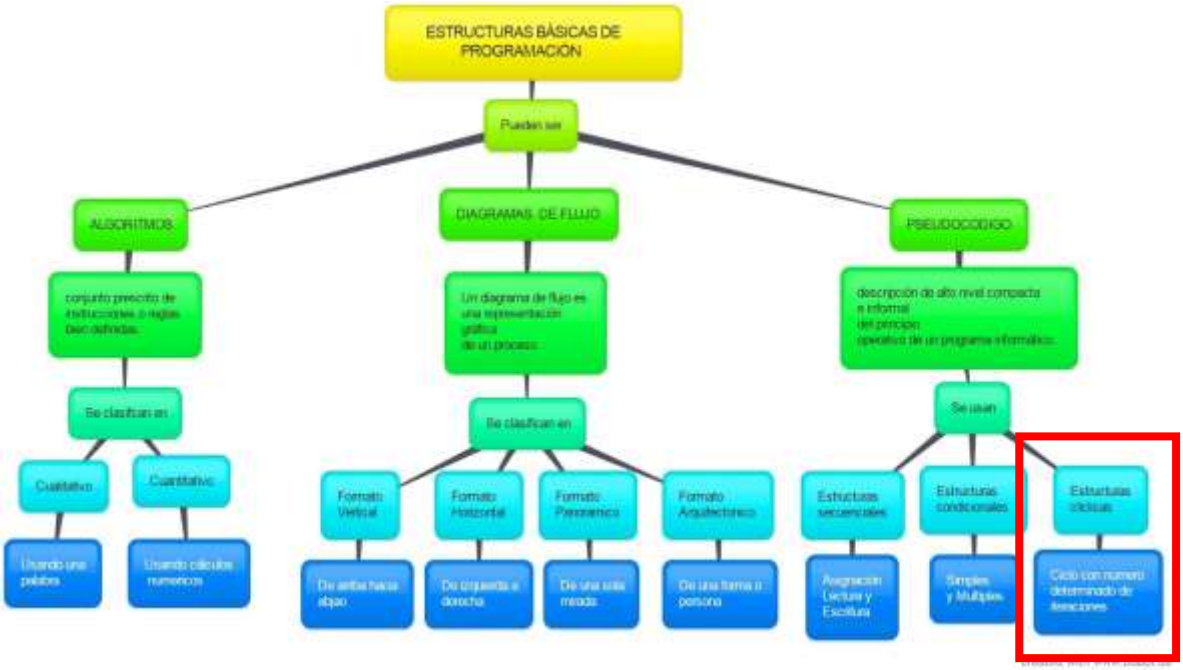
Objetivo: Realizar programas en Python, donde se visualicen estructuras cíclicas (for, while).

Herramienta de trabajo: Software

Python 3.9


MSDOS


Editor Visual Studio Code (Terminal),


ESTRUCTURAS BÁSICAS DE PROGRAMACION
Pueden ser
ALGORITMOS
DIAGRAMAS DE FLUJO
PSEUDOCODIGO
conjunto preciso de instrucciones o reglas bien definidas
Un diagrama de flujo es una representacion gráfica de un proceso
descripción de alto nivel compacta e informal del principio operativo de un programa informático
Se clasifican en
Se clasifican en
Se usan
Cualitativo
Cuantitativo
Formato Vertical
Formato Horizontal
Formato Panorámico
Formato Arquitectonico
Estructuras secuenciales
Estructuras condicionales
Estructuras cíclicas
Usando una palabra
Usando cálculos numéricos
De arriba hacia abajo
De izquierda a derecha
De una sola mirada
De una forma o persona
Asignacion Lectura y Escritura
Simples y Multiples
Ciclo con numero determinado de iteraciones

CICLOS EN PYTHON: PARA y MIENTRAS

Se utilizan los ciclos PARA (**for)** cuando conoces la cantidad de repeticiones y los ciclos MIENTRAS(**while)** cuando la cantidad de repeticiones depende de que se cumpla una condición.

Ciclo PARA: for

En muchas ocasiones se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle.

En estos casos en el que el número de iteraciones es fija, se debe usar la estructura desde o para. La estructura desde ejecuta las acciones del cuerpo del bucle un numero especificado de veces y de modo automático controla el número de iteraciones o pasos a través del cuerpo del bucle.

Como todos los bloques en Python en su primera línea finalizan con **:** luego las siguientes líneas deben llevar una tabulación o espacios en blanco para indicar que pertenecen al bloque. Los bloques for, a diferencia de los demás lenguajes de programación, funciona únicamente para recorrer una lista, entonces primer debemos contar con una lista.

En **Python no existe en tradicional** ciclo `for i=0; i<10; i++` en donde podíamos definir una cantidad de repeticiones, como lo mencione antes, solo puede recorrer una lista.

Entonces hay un truco que podemos usar para simular este comportamiento, solo debes usar la **función** **range** para crear una lista temporal.

```
for i in range(10):
    print(i)
```

Ejercicio 48_1: Imprimir los numero del 1 al 10.

ejercicio48_1.py > ...

```
1   # Ejercicio 48_1: Imprimir los numero del al 10
2
3   print("Ciclo de 0 a 10")
4   for i in range(11):
5       print(i)
```

Ejercicio 48_2: Imprimir los numero del 1 hasta el 5.

ejercicio48_2.py > ...

```
1   #Ejercicio 48_2: Imprimier los numero del 1 hasta el 5
2
3   print("Ciclo de 1 a 5")
4   for i in range(1, 6):
5       print(i)
6
7
```

Ejercicio 48_3:

ejercicio48_3.py > ...

```
1
2  # Ejericio 48_3: Imprimier los numero 1 hasta el 100, con intervalos de 10 y 50
3  print("ciclo de 10 a 100, en incrementos de 10")
4  for i in range(10, 110, 10):
5      print(i)
6
```

Ejercicio 48_4: Imprimir los numero del 1 al 10 descendiente….

ejercicio48_4.py > ...

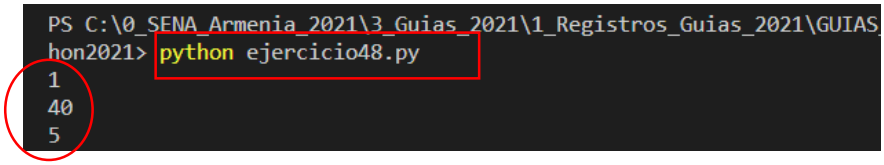
```
1  # Ejercicio 48_4: Imprimir los numero del 1 al 10 descendientes
2
3  print("ciclo de 10 a 1, usando incrementos negativos")
4  for i in range(10, 0, -1):
5      print(i)
6
7
```

VECTORES

Ejercicio 49: Declarar una variable llamada `vector` con (3) tres elementos, luego en el bloque `for` se declara una variable, en este caso se llama `i`, después uso la palabra reservada `in` y seguido de la lista (en este caso es `vector`)

```
1
2   vector = [1,40,5]
3
4   for i in vector:
5       print(i)
6
```

Terminal:



Ejercicio 50: Imprimir una lista de (4) nombres de personas.

```
vector = ["Maria", "Juan", "Pedro", "Felipe"]

for i in vector:
    print(i)
```

Ciclo Mientras: while

"Mientras haya algo que hacer hazlo".

El contenido de este bucle se ejecutará una y otra vez mientras el resultado de una expresión lógica sea 'Verdadero'.

Este ciclo se puede representar de la siguiente manera:

```
while expresión_condicional:
    instrucción
```

Sintaxis:

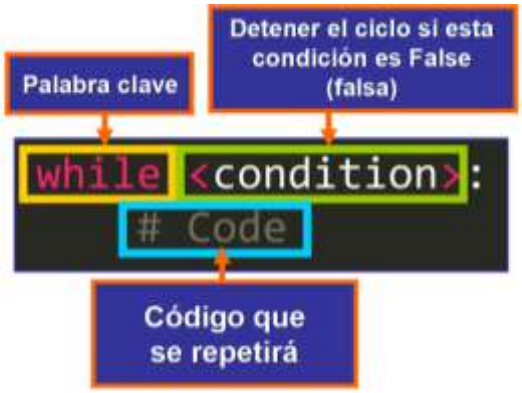
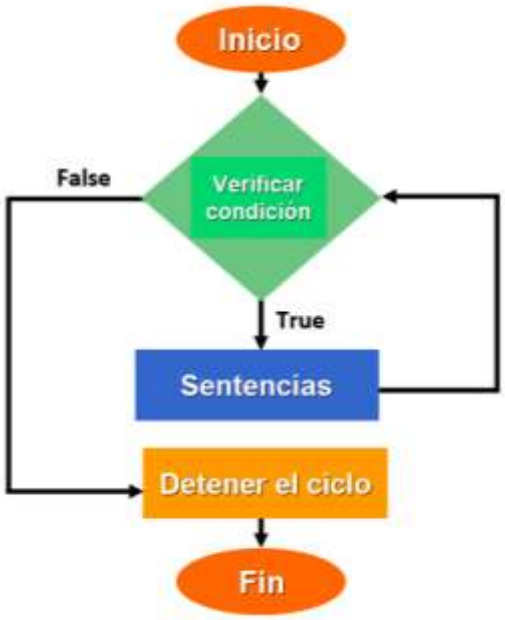


Diagrama - Ciclos while



¿Para qué se usan?. Los usamos para **repetir una secuencia de instrucciones o sentencias un número indefinido de veces**. Este tipo de ciclo se ejecuta **mientras** una condición dada es `True` (verdadera) y solo se detiene si la condición es `False` (falsa).

Dato: la palabra **`while`** significa "mientras" en español.

Cuando escribimos un ciclo while, no definimos explícitamente cuántas iteraciones serán completadas, solo escribimos una condición que debe ser verdadera (`True`) para continuar el proceso y falsa (`False`) para detenerlo.

Dato: si la condición del ciclo while nunca es falsa (`False`), entonces tendremos un ciclo infinito, el cual es un ciclo que en teoría nunca se detiene sin que sea interrumpido de forma externa. Por ejemplo, si se interrumpe con un atajo de teclado.

Ejercicio 51_1: Imprimir los numero del 1 hasta el 100, utilizando el ciclo Mientras (While)

ejercicio51_1.py > ...

```
1   #Ejercicio 51: Imprimir los numero del 1 hasta el 100,
2   # utilizando el ciclo Mientras (While)
3
4   i = 0
5   while i < 10:
6       # hacer_algo()
7       i +=1
8
9       print(i)
```

Tablas de multiplicas con los ciclos While y For

```
Digitar la tabla: 5
Tabla del: 5
 5 X 1 = 5
 5 X 2 = 10
 5 X 3 = 15
 5 X 4 = 20
 5 X 5 = 25
 5 X 6 = 30
 5 X 7 = 35
 5 X 8 = 40
 5 X 9 = 45
 5 X 10 = 50
```

Ejercicio 51_2: Tablas de Multiplicar – **CICLO MIENTRAS / While**

ejercicio51_2_tabla_mientras.py > ...

```python
#tabla de Multiplicar - Ciclo MIENTRAS / WHILE

tabla=int(input("Digitar la tabla: "))
print(f"Tabla del: {tabla}")

#Se inica el contador en cero
i=0
while (i<=9):
    i=i+1
    print(f" {tabla} X {i} = {tabla*i}")
```

Ejercicio 51_3: Tablas de Multiplicar – **CICLO PARA / For**

ejercicio51_3_tabla_para.py > ...

```python
#tabla de Multiplicar - Ciclo PARA / For

tabla=int(input("Digitar la tabla: "))
print(f"Tabla del: {tabla}")

#Se inica el contador en cero
i=0
while i<=9:
    i=i+1
    print(f" {tabla} X {i} = {tabla*i}")
```

Interrumpir ciclos

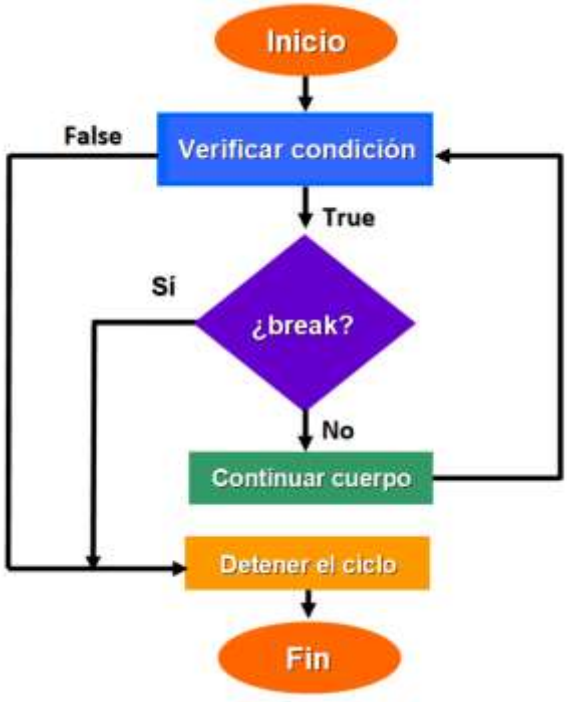
La sentencia "continue", al ejecutarlo, el programa se «salta» un ciclo o iteración y continúa con la siguiente iteración.

Ejercicio 52: Realizar un programa donde se ilustren los números del 1 al 10, saltándose el número 5, ver imagen.

```
1
2
3
4
6
7
8
9
10
```

```
for i in range(1, 11):
    if i == 5:
        continue
    print(i)
```

La sentencia break. Esta sentencia se usa para detener un ciclo while inmediatamente. Debes considerarla como una señal de tránsito roja de "stop" (detenerse) que puedes usar en tu código para controlar la funcionalidad del ciclo while.



Este es un ejemplo:

```
while True:
    # Código
    if <condición>:
        break
    # Código
```

El **comando break**, este comando interrumpe totalmente el ciclo, veamos el mismo ejemplo anterior, pero ahora usamos **break** en lugar de **continue.**

Ejercicio 53_1: Realizar un programa donde se ilustren los números del 1 al 10, saltándose el número 5, ver imagen.

```
for i in range(1, 10):
    if i == 5:
        break
    print(i)
```

El ciclo se corta al llegar al 5 y no imprime los demás números.

Ejercicio 53_2: Aquí tenemos un ejemplo de `break` en un ciclo `while True`:

```
while True:
    valor_ingresado = int(input("Ingrese un numero entero: "))
    if valor_ingresado % 2 != 0:
        print("Este numero es impar")
        break
    print("Este numero es par")

```

Ejercicio 54_1: Solicitar al usuario que ingrese una opción de un menú y después volvemos a pedir otra opción hasta que ingrese el texto “salir” y se salga del programa.

```
opcion = ''
while opcion != 'salir':
    opcion = input("Ingrese una opcion: ")
    print ("Usted eligio: " + opcion)
```

Ejercicios 54 hasta 67

Ejercicio 54_2: Realizar un programa donde se interrumpe el ciclo identificando si un número es impar o par.


```python
#Ejercicio 54_2: Realizar un programa donde se interrumpe el ciclo identificando
# si un número es impar o par.

while True:
    valor_ingresado=int(input("Ingresar un numero: "))

    if (valor_ingresado %2 !=0):
        print("El numero es impar")
        break
    print("El numero es par")
```

Ejercicio 55: Diseñar un algoritmo para determinar el promedio de notas de un estudiante. El programa de notas se finaliza cuando el usuario digite el numero–99.

Ejercicio 55_1

```python
contador = 0
suma = 0
nota = 1

while nota != -99:
    nota = float(input('Digite su nota (Digite -99 para terminar): '))
    if nota != -99:
        suma += nota
        contador +=1
        promedio= suma/contador
print(f"El promedio de sus notas es: {promedio}")
```

Ejercicio 55_2

ejercicio55.py > ...

```
1   i, nota, promedio = 0, 0, 0
2   while nota != -99:
3       nota = float(input("Ingrese la nota: "))
4       if nota == -99:
5           promedio = promedio / i
6           break
7       promedio = promedio + nota
8       i += 1
9   print(f"Promedio:{promedio}")
```

Ejercicio 56: Desarrollar el algoritmo que sume los números del 500 al 700.

Ejemplo.

Totalsuma = (500+501+502+503+504……+698+699+700**)**

Ejercicio 57: Un programa que sume los números ingresados por el usuario y cuando la suma sea superior a 100 deje de pedir números y muestre el total.

Ejercicio 58: Un programa que escoja entre dos opciones: **"1: hombre y 2: mujer".**

Si el usuario/a escoge opcion1, se mostrará el siguiente mensaje" ***Bienvenido!*** ", si escoge la opción 2, mostrará el mensaje" ***Bienvenida!*** ".

Pero si no escoge ninguna de las opciones deberá volver a pedir la opción hasta que ingrese una de las dos opciones.

Ejercicio 59: Un programa donde se determina la máxima y mínima temperatura de una lista de temperaturas, si es negativa se debe imprimir el total al finalizar el programa, el programa termina la entrada de datos con una temperatura 0.

// Sumatoria de las Temperaturas Negativas como las positivas.

Ejercicio 60: Un programa que permita la carga de 10 valores por teclado y nos muestre posteriormente la suma de los valores ingresados y su promedio.

Ejercicio 61: Mostrar los múltiplos de 8 hasta el valor 500. Debe aparecer en pantalla 8 - 16 - 24, etc.

Ejercicio 62: Al cerrar un expendio de naranjas, n clientes que aún no han pagado recibirán un 15% de descuento si compran más de 10 kilos. Determinar cuánto pagará cada cliente y cuanto percibirá la tienda por esas compras.

Ejercicio 63: Un zoológico cobra la entrada de admisión dependiendo de la edad del visitante:

- Los niños entre 3 y 12 años pagan $24
- Los adultos mayores de 65 pagaran $28
- El resto pagara $35.

Crea un programa que lea, una por una, las edades de un grupo de visitantes. El usuario debe introducir 0(cero) para indicar que ya terminó de capturar los datos. Una vez que el usuario haya terminado, el programa debe mostrar el total a pagar por el grupo.

Ejercicio 64: En un centro de verificación de automóviles se desea saber el promedio de puntos contaminantes de los n automóviles que lleguen.

Asimismo, se desea saber los puntos contaminantes del carro que menos contamino y del que más contamino.

Ejercicio 65: Realizar la tabla de multiplicar donde el usuario digite el número de la tabla, aplicando ciclos en la impresión.

Ejercicio 66: Un programa donde resuelve la siguiente serie donde el usuario digita X y n:

$$Ex = X^{1} + X^{2} + X^{3} + \ldots\ldots + X^{n}$$

Ejercicio 67: Crear un programa que pida al usuario un número y un símbolo, y dibuje un cuadrado usando ese símbolo.

El cuadrado tendrá el tamaño que ha indicado el usuario. Por ejemplo, si el usuario introduce 4 como tamaño y * como símbolo, deberá escribirse algo como:

```
**
**
**
**
```

Material de Referencia

- Cómo usar ciclos en Python, https://codigonaranja.com/ciclos-en-python
- https://www.freecodecamp.org/espanol/news/python-tutorial-ciclos-while/
- Ciclo Mientras, Video https://www.youtube.com/watch?v=nQxWthR71PI
-

Instructora. Carolina Zamaida González P. SENA