Programación

Introducción a la programación (Python)

Programación

Python

Programación

Editores on line

Editores on line:

- https://www.online-python.com/
- https://www.w3schools.com/python/trypython.asp?filename=demo_compiler
- https://pytwiddle.com

Programación

Entornos de desarrollo (IDE)

Entornos de desarrollo (IDE):

- https://www.python.org/
- https://www.jetbrains.com/es-es/pycharm/
- https://code.visualstudio.com/
- https://www.spyder-ide.org/
- https://jupyter.org/

Programación

Índice

- Variables
- Tipos de datos
- Imprimir.
- Leer.
- Operadores aritméticos.
- Operadores relacionales.
- Operadores lógicos.
- Otros operadores.
- Condicional simple.
- Condicional múltiple.
- Bucle while.
- Bucle for.
- Break y continue.

Programación

Variables (I)

- Una variable es algo parecido a un contenedor que tiene un nombre y almacena un valor.

```
nombre = “Juan Sin Miedo"
a = 5
total = 25.47
pagado = True
```

- Existen reglas para nombrar las variables.

Programación

Variables (II)

- Hay lenguajes en los que a las variables se les asigna un tipo de datos y no pueden almacenar datos de otro tipo.

Tipado estático	Tipado dinámico
int edad = 18 edad = 20 edad = “Juan Sin Miedo” (**Daría un error**)	edad = 18 edad = 20 edad = “Juan Sin Miedo”

- Python utiliza tipado dinámico.

Programación

Tipos de datos (I)

- Tipos de datos básicos en python:
  - Numéricos
    - int
    - float
    - complex
  - Booleano
    - bool
  - Cadenas de caracteres
    - str

https://docs.python.org/es/3/library/stdtypes.html
https://blog.hubspot.es/website/tipos-de-datos-python
https://ellibrodepython.com/numeros-complejos

```
#Cadena de caracteres
nombre = "Cadena de caracteres"
print (type(nombre))

#Entero
entero = 15
print (type(entero))

#Decimal
decimal = 15.75
print (type(decimal))

#Booleano
booleano = True
print (type(booleano))
```

Programación

Tipos de datos (II)

- Tipos de datos en python:
  - Secuencias
    - Listas
    - Tuplas
    - Rangos
  - Conjuntos
  - Diccionarios
  - Iteradores
  - Otros tipos de datos

https://docs.python.org/es/3/library/stdtypes.html

https://blog.hubspot.es/website/tipos-de-datos-python

Programación

Imprimir (I)

```
print (“Hola mundo”)
print (“Hola mundo”)
print (‘Hola mundo’)
print (“Hola, hoy es ‘viernes’ todo el día”)
print (‘Hola, hoy es “viernes” todo el día’)
print ()
print (“Hola mundo”)
print (“Hola”, “mundo”)
print (“Hola”, end=“ ”)
print (“mundo”)
```

Se pueden utilizar comillas dobles o comillas simples

Programación

Imprimir (II)

```
print (“Esta es la primera línea.”)
print (“Esta es la segunda línea”, end=“ “)
print (“y esta sigue siendo la segunda línea.”)
print (“Esta es la tercera línea”, end=“ “)
print (“y esta sigue siendo la tercera línea”, end=“ “)
print (“y sigue siendo la tercera línea.”)
print (“Esta es la cuarta línea.”)
print()
mensaje = “””Este es un mensaje
que se escribe en varias líneas.
Esta es la última.”””
print (mensaje)
```

El argumento **end** sirve para indicar la cadena que se imprime al final de la línea de argumentos. Por defecto es un salto de línea (end='\n'), pero puede ser cualquier otra cadena.

Se puede escribir en varias líneas utilizando tres comillas dobles

Programación

Secuencias de escape

```
print ("Esta es la primera línea.\nEsta es la segunda línea.")
print ("Esta línea esta alineada a la izquierda.")
print ("\tEsta línea esta tabulada hacia dentro.")
print ("\t\tEsta línea esta dos veces tabulada hacia dentro.")
print ("Esta línea esta alineada a la izquierda.")
print ("\nEsta es la última línea.")
```

Secuencia de escape	Valor
\n	Nueva línea
\t	Tabulador
\r	Retorno de carro
\"	Comilla doble
\'	Comilla simple

Programación

Ejercicios

Ejercicio 1:
Escribir un programa que muestre un mensaje de tres líneas por pantalla.

Ejercicio 2:
Escribir un programa que dibuje una figura utilizando el símbolo “*”

Ejercicio 3:
Escribir un programa que muestre un menú con cinco opciones por pantalla.

Ejercicio 4:
Crear un programa que muestre los datos personales del alumno (nombre y apellidos, lugar donde vive, centro en el que estudia, etc.) y un saludo.

Programación

Leer

```
#Leer una cadena de caracteres
cadena = input ("Introduzca una cadena de caracteres:  ")
print (type (cadena))

#Leer un número entero
entero = int (input("Introduzca un número entero:  "))
print (type (entero))

#Leer un número decimal
decimal = float (input ("Introduzca un número decimal:  "))
print (type (decimal))

#Leer un número booleano
booleano = bool (input ("Introduzca un valor booleano (True/False):  "))
print (type (booleano))
```

Programación

Operadores aritméticos (I)

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Potencia (**)
- División entera (//)
- Módulo o resto (%)

```
num01 = 5
num02 = 4
print (num01 + num02)
print (num01 - num02)
print (num01 * num02)
print (num01 ** num02)
print (num01 / num02)
print (num01 // num02)
print (num01 % num02)
```

https://www.freecodecamp.org/espanol/news/operadores-basicos-en-python-con-ejemplos/

Programación

Operadores aritméticos (II)

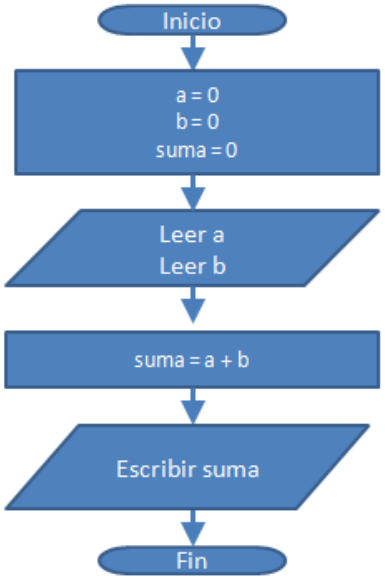
```
num01 = 5
num02 = 4
suma = 0
suma = num01 + num02
print (suma)
```

```
num01 = 5
num02 = 4
print (num01 + num02)
```

```
num01 = 5
num02 = 4
suma = 0
suma = num01 + num02
print ("La suma de " + str(num01) + " y " + str(num02) + " es igual a " + str(suma))
```

Programación

Operadores aritméticos (III)

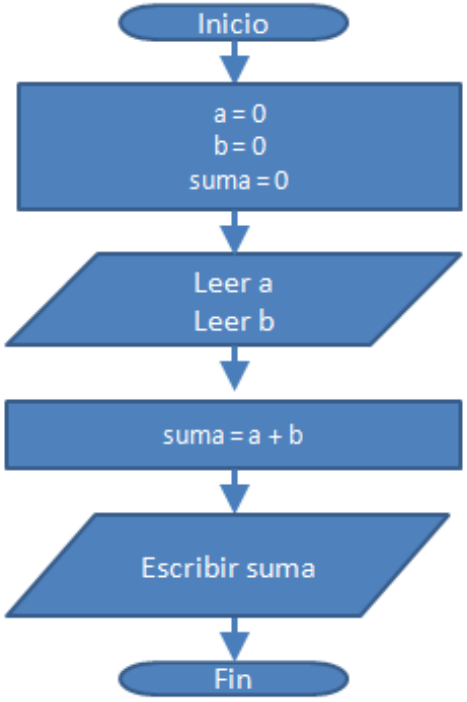



```
#Pedir datos
a = int (input (“Introduzca el primer número entero:  ”))
b = int (input(“Introduzca el segundo número entero:  ”))
#Procesar datos
suma = a + b
#Mostrar el resultado
print (suma)
```

```
Inicio
    leer (a)
    leer (b)
    suma = a + b
    escribir (suma)
Fin
```

Programación

Operadores aritméticos (IV)




```
#Pedir datos
a = int (input ("Introduzca el primer número entero:  "))
b = int (input ("Introduzca el segundo número entero:  "))
#Procesar datos
suma = a + b
#Mostrar el resultado
print ("La suma de", a, "y", b, "es igual a", suma)
```

https://www.online-python.com/

```
Inicio
    leer (a)
    leer (b)
    suma = a + b
    escribir (suma)
Fin
```

Programación

Precedencia de operadores

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x` `-x` `~x`	Unary plus, unary minus, and bitwise NOT
`*` `/` `//` `%`	Multiplication, division, floor division, and modulus
`+` `-`	Addition and subtraction
`<<` `>>`	Bitwise left and right shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`==` `!=` `>` `>=` `<` `<=` `is` `is not` `in` `not in`	Comparisons, identity, and membership operators
`not`	Logical NOT
`and`	AND
`or`	OR

If two operators have the same precedence, the expression is evaluated from left to right.

Programación

Ejercicios

Ejercicio 1:
Escribe un programa que calcule la suma de 3, 5 y 12.
Ejercicio 2:
Escribe un programa que calcule el producto de 5 y 15.
Ejercicio 3:
Escribe un programa que calcule el resultado de elevar 2 al cubo.
Ejercicio 4:
Escribe un programa que calcule el la división de 25 entre 4.
Ejercicio 5:
Escribe un programa que calcule el cociente y el resto de dividir 25 entre 4.

Programación

Operadores relacionales

- Mayor (>)
- Menor (<)
- Mayor o igual (>=)
- Menor o igual (<=)
- Igual (==)
- Distinto (!=)

```
num01 = 5
num02 = 4
print (num01 > num02)
print (num01 < num02)
print (num01 >= num02)
print (num01 <= num02)
print (num01 == num02)
print (num01 != num02)
```

https://www.freecodecamp.org/espanol/news/operadores-basicos-en-python-con-ejemplos/

Programación

Operadores lógicos

- And (and)
- Or (or)
- Not (not)

```
print (False and False)
print (False and True)
print (True and False)
print (True and True)
print()
print (False or False)
print (False or True)
print (True or False)
print (True or True)
print()
print (not False)
print (not True)
```

https://www.freecodecamp.org/espanol/news/operadores-basicos-en-python-con-ejemplos/

Programación

Precedencia de operadores

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x` `-x` `~x`	Unary plus, unary minus, and bitwise NOT
`*` `/` `//` `%`	Multiplication, division, floor division, and modulus
`+` `-`	Addition and subtraction
`<<` `>>`	Bitwise left and right shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`==` `!=` `>` `>=` `<` `<=` `is` `is not` `in` `not in`	Comparisons, identity, and membership operators
`not`	Logical NOT
`and`	AND
`or`	OR

If two operators have the same precedence, the expression is evaluated from left to right.

Programación

Otros operadores

- in (not in)
- is (not is)

```
numeros = [1, 3, 5, 7, 9]
print(numeros)
a = 2
if a in numeros:
        print("El número " + str(a) + " está en la lista")
else:
        print("El número " + str(a) + " no está en la lista")
```

```
numero = int(input("Introduce un número entero: "))
#numero = input("Introduce un número entero: ")
if type(numero) is int:
        print(numero *2)
else:
        print("El número introducido no es un entero")
```

https://www.freecodecamp.org/espanol/news/operadores-basicos-en-python-con-ejemplos/

Programación

in (not in)

```
numeros = [1, 3, 5, 7, 9]
print(numeros)
a = 2
if a in numeros:
        print("El número " + str(a) + " está en la lista")
else:
        print("El número " + str(a) + " no está en la lista")
```

https://www.freecodecamp.org/espanol/news/operadores-basicos-en-python-con-ejemplos/

Programación

is (not is)

```
numero = int(input("Introduce un número entero: "))
#numero = input("Introduce un número entero: ")
if type(numero) is int:
        print(numero *2)
else:
        print("El número introducido no es un entero")
```

https://www.freecodecamp.org/espanol/news/operadores-basicos-en-python-con-ejemplos/

Programación

Precedencia de operadores

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x` `-x` `~x`	Unary plus, unary minus, and bitwise NOT
`*` `/` `//` `%`	Multiplication, division, floor division, and modulus
`+` `-`	Addition and subtraction
`<<` `>>`	Bitwise left and right shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`==` `!=` `>` `>=` `<` `<=` `is` `is not` `in` `not in`	Comparisons, identity, and membership operators
`not`	Logical NOT
`and`	AND
`or`	OR

If two operators have the same precedence, the expression is evaluated from left to right.

Programación

Ejercicios

Ejercicio 1:
Escribe un programa que pida al usuario dos números enteros y muestre por pantalla el resultado de la suma, la resta, la multiplicación y la división de esos dos números. El resultado de cada operación debe mostrarse en una línea diferente.

Ejercicio 2:
Escribe un programa que permita calcular el perímetro y el rea de un cuadrado. El valor del lado ha de ser introducido por el usuario.

Ejercicio 3:
Escribe un programa que permita calcular el volumen de un prisma rectangular. Las longitudes de las aristas han de ser introducidas por el usuario.

Programación

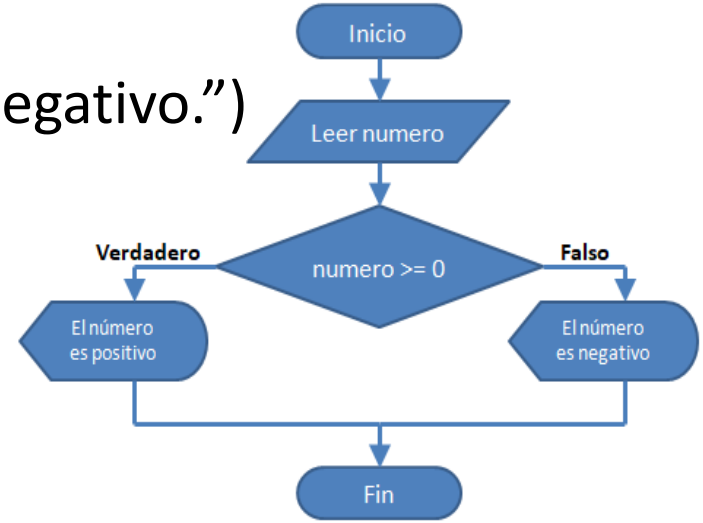
Condicional simple (I)

```
#Pedir datos
numero = int (input (“Introduzca un número entero:  ”))
#Procesar datos y mostrar el resultado
if numero >=0:
        print (“El número es positivo.”)

print (“Fin de la aplicación.”)
```

Programación

Condicional simple (II)

```
#Pedir datos
numero = int (print (“Introduzca un número entero: ”))
#Procesar datos y mostrar el resultado
if numero >=0:
        System.out.println (“El número es positivo.”)
else:
        System.out.println (“El número es negativo.”)

print (“Fin de la aplicación.”)
```



Programación

Condicional múltiple (I)

```
#Pedir datos
numero = int ( input (“Introduzca un número entero:  ”))
#Procesar datos y mostrar el resultado
if numero > 0:
        print (“El número es positivo.”)
elif numero == 0:
        print (“El número es igual a cero.”)
else:
        print (“El número es negativo.”)
```

Programación

Condicional múltiple (II)

```
nota=float(input("Introduce la nota numérica: "))
if nota<5:
        print("Su nota es Insuficiente")
elif nota<6:
        print("Su nota es Suficiente")
elif nota<7:
        print("Su nota es Bien")
elif nota<9:
        print("Su nota es Notable")
elif nota<=10:
        print("Su nota es Sobresaliente")
else:
        print("La nota ha de estar comprendida entre 0 y 10")
```

Programación

Ejercicios

Ejercicio 1:
Escribe un programa que pida la edad al usuario y diga si es mayor o menor de edad.

Ejercicio 2:
Escribe un programa que pida un número al usuario y diga si ese número es par o impar.

Ejercicio 3:
Escribe un programa que pida al usuario un número del 1 al 7 y diga a que día de la semana corresponde.

Ejercicio 4:
Escribe un programa que pida al usuario un número del 1 al 12 y diga a que mes corresponde.

Programación

Ejercicios

Ejercicio 5:
Escribe un programa que pida un número al usuario comprendido entre 0 y 100 y diga a que cuartil corresponde.

Ejercicio 6:
Escribe un programa que realice la división de dos números comprobando que el divisor no es cero.

Programación

Bucle while (I)

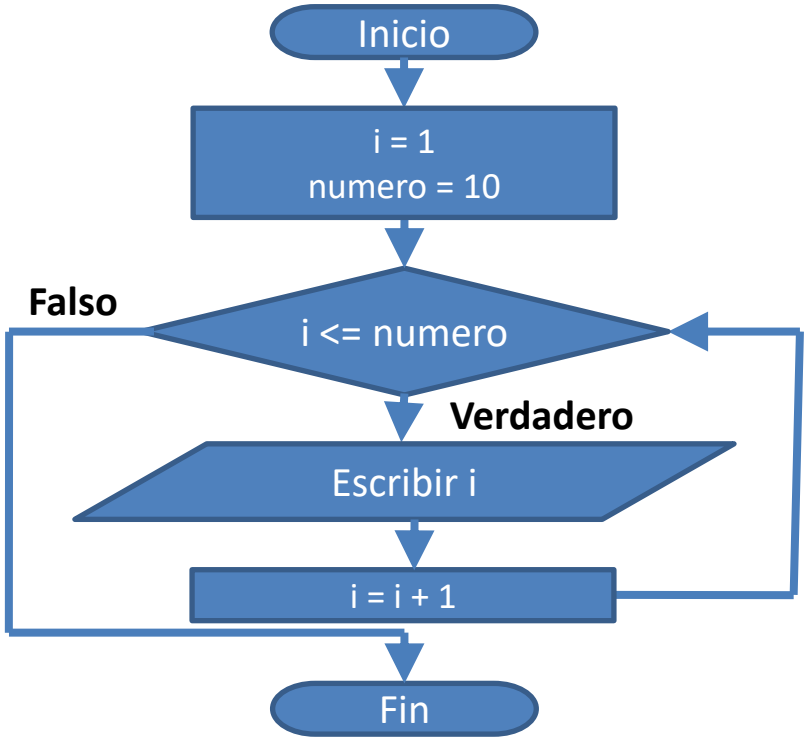
```
#Escribir 10 veces una frase
i = 1
while i <= 10:
        print (“Hoy es un gran día.”)
        i=i+1
```

```
#Pedir datos
veces = int (input (“Introduzca un número entero:  ”))
#Escribir n veces una frase
i = 1
while i <= veces:
        print ("Hoy es un gran día.")
        i=i+1
```

Programación

Bucle while (II)

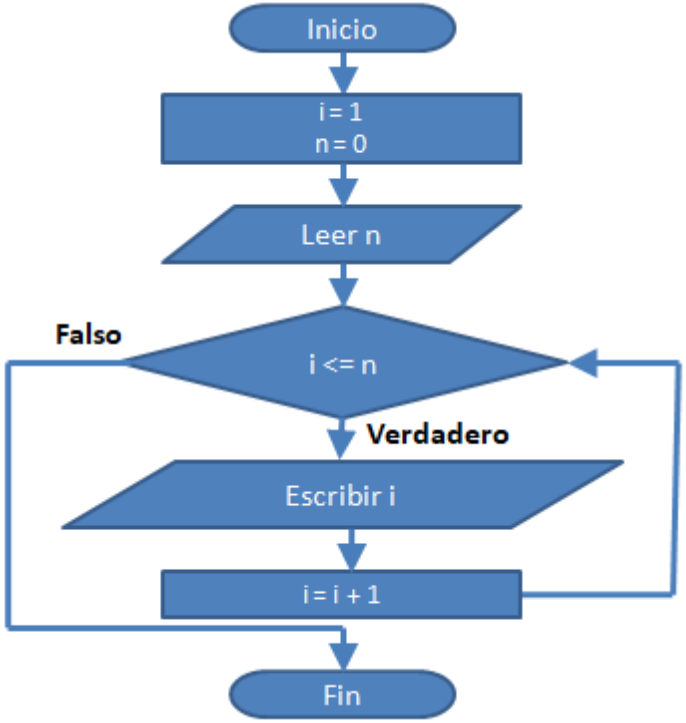
```
#Pedir datos
numero = 10
#Contar desde 1 hasta numero
i = 1
while i <= numero:
        print ( i )
        i=i+1
```



Programación

Bucle while (III)

```
#Pedir datos
numero = int (input (“Introduzca un número entero:  ”))
#Contar desde 1 hasta numero
i = 1
while i <= numero:
        print ( i )
        i=i+1
```



Programación

Bucle while (IV)

```
#Programa que calcula la suma de los n primeros números naturales
#Pedir datos
print (“Introduzca un número entero:  ”)
numero = int (input())
#Procesar datos
suma=0
i = 1
while  i <= numero:
        suma = suma + i
        i = i + 1
 #Mostrar el resultado
print (suma)
```

Programación

Bucle for (I)

```
#Bucle for
#Imprimir los números del 0 al 9
for i in range (0, 10):
        print ( i )
```

```
#Bucle for
# Imprimir los números del 0 al 10
for i in range (0, 11):
        print ( i )
```

Programación

Bucle for (II)

```
#Imprimir los números del 1 al 10
for i in range (0, 10):
        print ( i + 1)
```

```
numero= 10;
#Imprimir los números del 1 al 10
for i in range (0, numero):
        print ( i + 1)
```

Programación

Bucle for (III)

```
#Bucle for
for i in range (0, 10):
        print ( i )
```

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

```
#Bucle for
for i in range (0, 10, 2):
        print ( i )
```

```
#Bucle for
for i in range (0, 10, 3):
        print ( i )
```

Programación

Bucle for (IV)

```
#Pedir datos
numero = int (input (“Introduzca un número entero:  ”))
#Procesar datos y mostrar el resultado
for i in range (0, numero):
        print ( i + 1)
```

Programación

Bucle for (V)

```
#Pedir datos
print (“Introduzca un número entero:  ”)
numero = int (input())
#Procesar datos
suma=0
for i in range (1, numero +1):
        suma = suma + i
#Mostrar el resultado
print (suma)
```

Programación

Equivalencia bucles while y for

```
suma = 0
i=1
while i<=100:
    suma = suma + i
    i = i + 1
print("El resultado es: " + str(suma))
```

```
suma = 0
for i in range (1, 101):
    suma = suma + i
print("El resultado es: " + str(suma))
```

Programación

Ejercicios

Ejercicio 1:
Escribe un programa que cuente hasta un valor introducido por el usuario.

Ejercicio 2:
Escribe un programa que pide al usuario dos números y muestre los valores comprendidos entre ellos.

Ejercicio 3:
Escribe un programa que pida al usuario dos números y muestre el resultado de la suma por pantalla. El proceso se ha de repetir hasta que el usuario introduzca un cero.

Programación

Ejercicios

Ejercicio 4:
Escribe un programa que pida diez números al usuario y muestre por pantalla la suma y el producto de esos números.

Ejercicio 5:
Escribe un programa que calcule los divisores de un número introducido por el usuario.

Ejercicio 6:
Escribe un programa que pida números al usuario hasta que se introduzca un número negativo y muestre por pantalla la cantidad de números introducidos.

break y continue

Programación

break y continue (I)

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

La instrucción **break** permite salir de un bucle cuando se cumple una determinada condición

La instrucción **continue** permite abandonar la iteración actual y pasar a la siguiente iteración de un bucle

Programación

pass

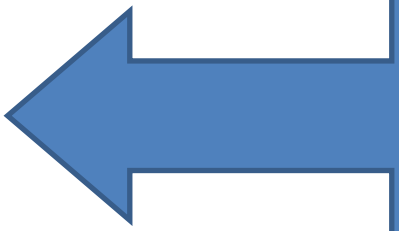
```
for i in range(10):
        if i == 5:
                continue
        print("El número es " + str( i ))
print("Fin del bucle")
print()
for i in range(10):
        if i == 5:
                pass
        print("El número es " + str( i ))
print("Fin del bucle")
```

La instrucción **pass** se utiliza cuando, por diferentes razones (se quiere agregar el código más tarde, se quieren ignorar algunas excepciones que se generan durante el tiempo de ejecución, etc.), el usuario no quiere que el programa haga nada

Programación

pass

```
x = 2
if x == 2:
        pass
else:
        print ("La x no vale 2")
print ("Fin")
```



La instrucción **pass** se utiliza cuando, por diferentes razones (se quiere agregar el código más tarde, se quieren ignorar algunas excepciones que se generan durante el tiempo de ejecución, etc.), el usuario no quiere que el programa haga nada

Fin

Programación

Introducción a la programación (Python)