

# FUNDAMENTOS DE PYTHON

---

JORGE VILLAVICENCIO



# PRIMERO

## Introducción a Python

# Presentación del docente

Ahora, conoceremos algunos aspectos relevantes del experto y del contenido a abordar en este curso.

Los temas que abordaremos son:

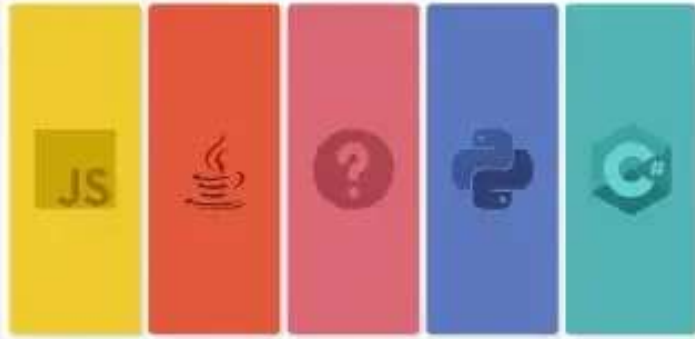
- ¿Quién soy?
- ¿Qué hago? Mi experiencia
- ¿Por qué es importante llevar este curso?
- ¿Qué aprenderán en este curso?



# JORGE VILLAVICENCIO

## ¿Quién soy?

- Magíster en Ciencias de la Computación con especialidad en Inteligencia Artificial por la Universidad de Ingeniería y Tecnología de Perú.
- Ingeniero de Software de la Universidad Nacional Mayor de San Marcos con amplia experiencia en desarrollo de software, calidad de software y gestión de proyectos en metodologías ágiles.
- Mis áreas principales de interés son la ingeniería de software, calidad y pruebas de software e inteligencia artificial.



## ¿Qué hago?

- Actualmente soy profesor de laboratorio de programación I y II en la universidad UTEC.
- Pertenezco al proyecto VINCI, el cual tiene como finalidad la creación del primer registro genómico peruano utilizando inteligencia artificial.



## ¿Por qué es importante llevar este curso?


- Conocerás los fundamentos básicos de Python.
- Comprenderás la diferencia entre un algoritmo, programa y lenguaje de programación.
- Podrás automatizar actividades diarias utilizando condicionales o repetitivas.



```
print ("Hello, World!")
```

## ¿Qué aprenderán en este curso?

Los fundamentos básicos de Python, desde la importancia de construir algoritmos y la creación de variables y expresiones hasta el uso de estructuras de control selectivas (if else) y repetitivas (while y for).

A black and white photograph of Steve Jobs, smiling and gesturing with his hands. He is wearing his signature round glasses and a dark turtleneck. The background is dark, with a blurred light source on the left.

**“No tenemos la oportunidad de hacer muchas cosas, por lo que cada una de ellas debería ser excelente”**

---

Steve Jobs



# Importancia de la programación

Ahora, comprenderemos la importancia de Python dentro de la programación.

Los temas que abordaremos son:

- ¿Qué es un algoritmo?
- ¿Qué es un programa?
- ¿Qué es un lenguaje de programación?
- ¿Por qué Python?

# ¿Qué es un algoritmo?

Un algoritmo es un conjunto de instrucciones diseñadas para realizar una tarea específica.

Un algoritmo es una lista finita de instrucciones que se utiliza con mayor frecuencia para resolver problemas o realizar tareas.

## Calcular el número mayor

**Entrada (input)**



1. Ingresar el número X
2. Ingresar el número Y
3. Si X es mayor al número Y  
El número mayor es X
4. Sino Y es el número mayor



**Salida (output)**

# ¿Qué es un programa?

Un programa es una secuencia de instrucciones escritas para realizar una tarea específica en una computadora.

```
1  x = int(input("Ingresar número 1"))
2  y = int(input("Ingresar número 2"))
3  if x > y:
4      |   print("{} es el número mayor".format(x))
5  else:
6      |   print("{} es el número mayor".format(y))
```

# ¿Qué es un lenguaje de programación?

Es la forma como nos comunicamos con una computadora a través de instrucciones escritas en una sintaxis.



# ¿Qué es un lenguaje de programación?

## Algoritmo

Calcular el número mayor

1. Ingresar el número X
2. Ingresar el número Y
3. Si X es mayor al número Y  
El número mayor es X
4. Sino Y es el número mayor

## Programa

```
1 x = int(input("Ingresar número 1"))
2 y = int(input("Ingresar número 2"))
3 if x > y:
4     print("{} es el número mayor".format(x))
5 else:
6     print("{} es el número mayor".format(y))
```

## Lenguaje de programación



# ¿Por qué Python?

- Es uno de los primeros lenguajes de programación más utilizados.
- Código abierto.
- Multipropósito (desarrollo consola y web).
- Librerías y frameworks para facilitar el desarrollo.
- Permite automatizar procesos.
- Las empresas actuales solicitan conocerlo como lenguaje de programación.

```
1 2 3 4 5 6 7 8 9 10 11
2 3 4 5 6 7 8 9 10 11 12
3 4 5 6 7 8 9 10 11 12 13
```

“

**print(“Hola, Bienvenidos  
al curso de Python”)**

”

```
1 2 3 4 5 6 7 8 9 10 11
2 3 4 5 6 7 8 9 10 11 12
3 4 5 6 7 8 9 10 11 12 13
```

# Configuración del ambiente de desarrollo

Ahora, identificaremos los pasos para instalar el ambiente de desarrollo.

Los temas que abordaremos son:

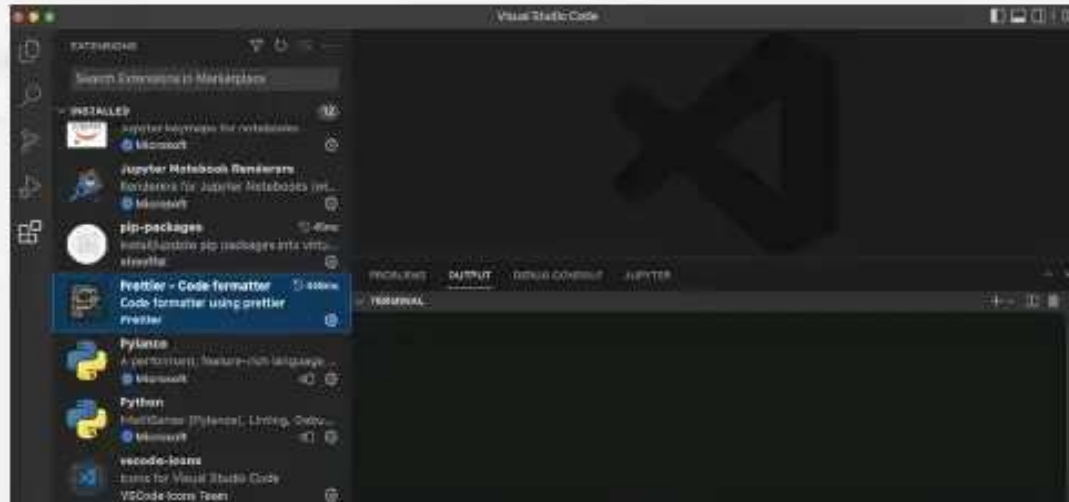
- Instalación de Python compilador
- Instalación de Visual Code
- Conociendo Visual Code
- Alternativas de IDE de desarrollo



# Instalación de Python compilador

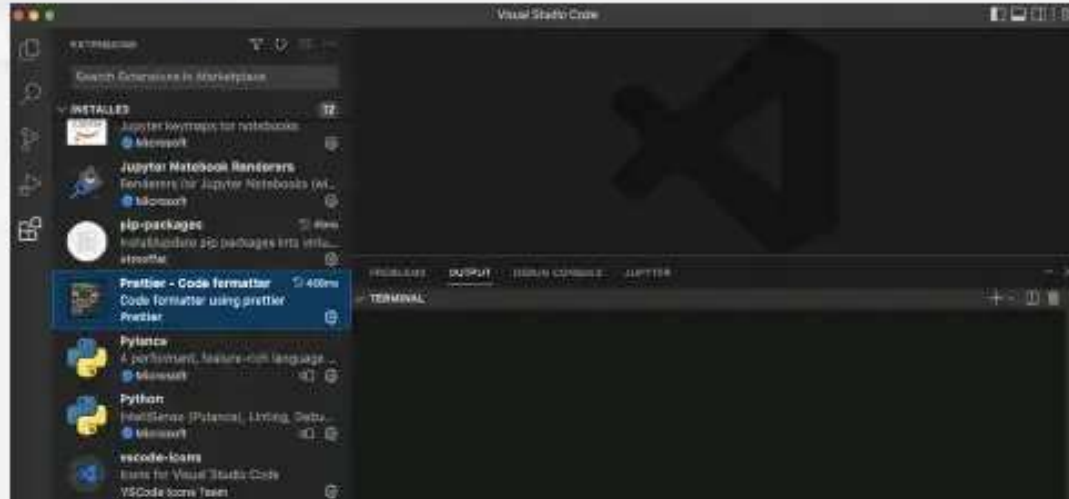


# Instalación de Visual Code



IDE o entorno de desarrollo integrado

# Conociendo Visual Code



# **Alternativas de IDE de desarrollo**



# Alternativas de IDE de desarrollo





**print(“Visual Code, nuestro  
nuevo mejor amigo”)**



# Ideas resumen del módulo 1

01

Un algoritmo es una lista finita de instrucciones que se utiliza con mayor frecuencia para resolver problemas o realizar tareas.

02

Un lenguaje de programación es la forma como nos comunicamos con una computadora a través de instrucciones escritas en una sintaxis.

03

Visual Code es el ambiente de desarrollo integrado para el curso de fundamentos de Python.



# BIODUNLOO

Conceptos básicos  
de Python



# Variables y expresiones

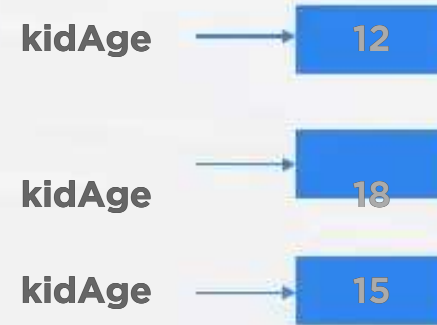
Ahora, conoceremos los componentes básicos para armar una expresión en Python.

Los temas que abordaremos son:

- Variables
- Tipos de datos básicos
- Operador de asignación
- Operadores aritméticos
- Operadores de relación

# Variables

- Un **valor con nombre** que, potencialmente, se puede cambiar a medida que se ejecuta el programa.
- La variable se escribe siempre a la izquierda de la igualdad.
- El nombre de una variable no puede contener espacios.
- No se pueden utilizar las palabras reservadas como variables (def, list, dict, etc.).



# Tipos de datos

## Numéricos

Tipo	Nombre	Descripción	Ejemplo
Enteros	int	Representan los números Enteros (positivos y negativos)	12 0 -5
Flotantes	float	Son todos los números que contienen dígitos después de un punto decimal	13.4 0.4 -0.12

# Tipos de datos

## Lógicos

Tipo	Nombre	Descripción	Ejemplo
Booleanos	bool	Representan los valores de verdad (Verdadero o falso)	True False

# Tipos de datos

## Cadena de caracteres (String)

Tipo	Nombre	Descripción	Ejemplo
Cadenas	str	Expresiones formadas por caracteres.  Se pueden representar con <b>comillas simples</b> o <b>dobles</b>	"Hola mundo" 'Hola mundo'  "A" 'AA'

# Operador de asignación

Operador	Nombre	Ejemplo
=	Asignación	x = y

# Operadores aritméticos

Operador	Nombre	Ejemplo
+	Adición	$x + y$
-	Sustracción	$x - y$
*	Multiplicación	$x * y$
/	División	$x / y$
%	Módulo	$x \% y$
**	Exponenciación	$x ** y$
//	División de piso	$x // y$

# Operadores de relación

Operador	Nombre	Ejemplo
==	Iguales	x == y
!=	Diferentes	x != y
<	Mayor	x < y
>	Menor	x > y
<=	Mayor igual	x <= y
=>	Menor igual	x => y





“**print(“Variables”)**”

# Estructuras de control: Selectivas I

Ahora entenderemos cómo utilizar las estructuras de control selectiva.

Los temas que abordaremos son:

- ¿Qué es una estructura de control selectiva?
- Selectivas simples
- Selectivas dobles

# ¿Qué es una estructura de control selectiva?



# Selectiva Simple

## Algoritmo

**1. Si** la condición es verdadera:

1. Instrucción 1
2. Instrucción 2
3. ...
4. Instrucción x

## Programa en Python

```
1 if condition:  
2     instruction_1  
  
3     instruction_2  
4     ...  
5     instruction_x
```

# Selectiva simple

## Ejemplo

Crear un programa en Python que permita ingresar la edad de una persona e indicar si es mayor de edad.

Edad	$n \geq 18$	Resultado
18	True	Mayor de edad
27	True	Mayor de edad
5	False	
10	False	

# Selectiva doble

## Algoritmo

### 1. Si la condición es verdadera:

1. Instrucción 1
2. Instrucción 2
3. ...
4. Instrucción x

### 2. En caso contrario:

1. Instrucción 1
2. Instrucción 2
3. ...
4. Instrucción y

## Programa en Python

```
1  if condition:
2      instruction_1
3      instruction_2
4      ...
5      instruction_x
6  else:
7      #instruction_1
8      #instruction_2
9      #...
10     #instruction_y
```

# Selectiva doble

## Ejemplo

Crear un programa en Python que permita ingresar la edad de una persona e indicar si es mayor de edad o menor de edad.

Edad	n >= 18	Resultado
18	True	Mayor de edad
27	True	Mayor de edad
5	False	Menor de edad
10	False	Menor de edad



“

**decisión = input(“Decisión  
1 o 2, tú decides”)**

”



# Estructuras de control: Selectivas II

Ahora, entenderemos cómo utilizar las estructuras de control selectiva.

Los temas que abordaremos son:

- Selectivas múltiples
- Selectivas anidadas

# Selectivas múltiples

## Algoritmo

1. Si la c1 es verdadera:

1. Instrucción 1
2. Instrucción 2
3. ...
4. Instrucción x

2. Caso contrario y c2 es verdad:

1. Instrucción 1
3. ...
4. Instrucción y

3. Caso contrario a los anteriores:

1. ...

## Programa en Python

```
1 if condition:
2     instruction_1
3     instruction_2
4     ...
5     instruction_x
6 elif condition2:
7     #instruction_1
8     #instruction_2
9     #...
10    #instruction_y
else:
    #instructions
```

# Selectivas múltiples

## Ejemplo

Crear un programa en Python que permita ingresar un número e indicar si es positivo, negativo o cero.

n	$n > 0$	$n < 0$	Resultado
-5	False	True	Ingresaste un número negativo
-1	False	True	Ingresaste un número negativo
0	False	False	
5	True		Ingresaste un número positivo

# Selectivas anidadas

Crear un programa en Python que permita ingresar la edad de una persona e indicar si es mayor de edad o menor de edad. **Si es menor de edad, validar si el valor de la edad es mayor a cero, sino indicar que la edad no es correcta.**

```
1  edad = int(input("Ingresar edad: "))
2  if edad >= 18:
3      print("Mayor de edad")
4  else:
5      if edad > 0:
6          print("Menor de edad")
7      else:
8          print("Edad no correcta")
```



“

**decisión = input(“Decisión  
1 o 2, tú decides”)**

”

## Ideas resumen del módulo 2

**01**

Una variable es un valor con nombre que puede ir cambiando durante toda la ejecución del programa.

**02**

Existen los operadores de asignación, aritméticos y de relaciones.

**03**

Utilizamos las estructuras de control selectivas para tomar la decisión de si una condición es verdadera o falsa.



# MÓDULO

Estructuras de  
control: Repetitivas

# Estructura While

Ahora entenderemos cómo utilizar la estructura de control repetitiva while.

Los temas que abordaremos son:

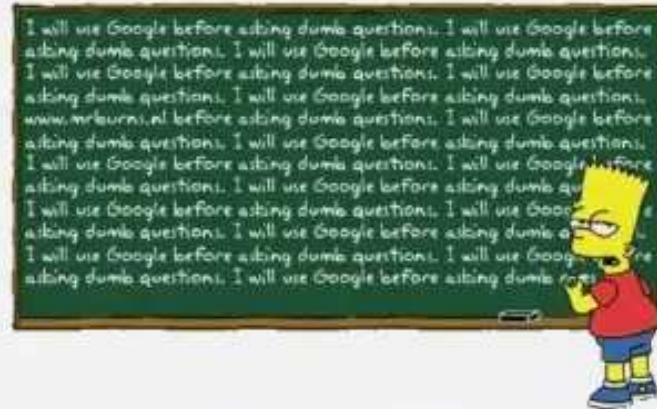
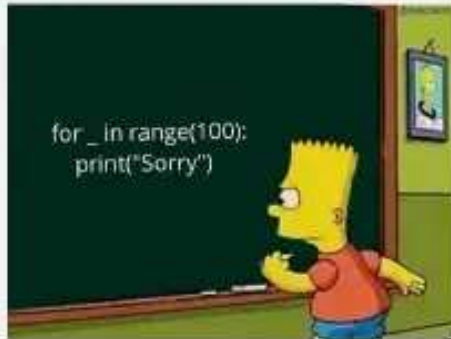
- ¿Qué es una estructura de control repetitiva?
- Uso de la estructura de control While
- While anidado



# ¿Qué es una estructura de control repetitiva?

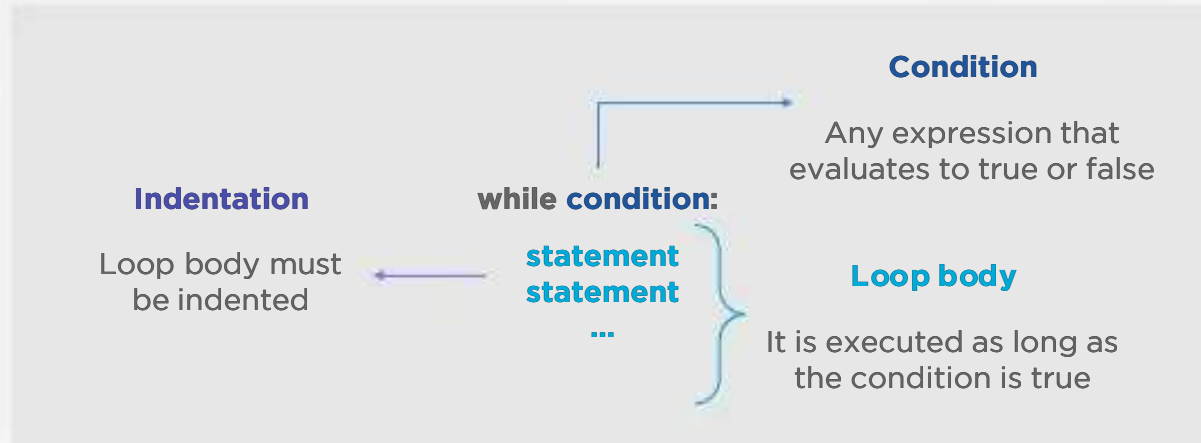
Teacher: Write "Sorry" 100 times

Me:



# Uso de la estructura de control While

Un bucle while permite repetir la ejecución de un grupo de instrucción mientras se cumpla una condición (es decir, mientras la condición tenga el valor True)



# While anidado

Un While anidado es cuando dentro de un bucle while definimos 1 o más while.

## Ejemplo

Crear un programa que permita ingresar los nombres de N alumnos y la cantidad M de cursos por alumno.

```
3  N = int(input("Ingresar cantidad de estudiantes: "))
4  i = 1
5  j = 0
6  while N >= i: # 10 != 10 Falso
7      nombre = input("ALUMNO {}: ".format(i))
8      M = int(input("Ingresar cantidad de cursos: "))
9      while M > j:
10         curso_nombre = input("CURSO {}: ".format(j+1))
11
12     i += 1
```



**Un While para repetir acciones  
facilita nuestro día a día**



# Estructura For

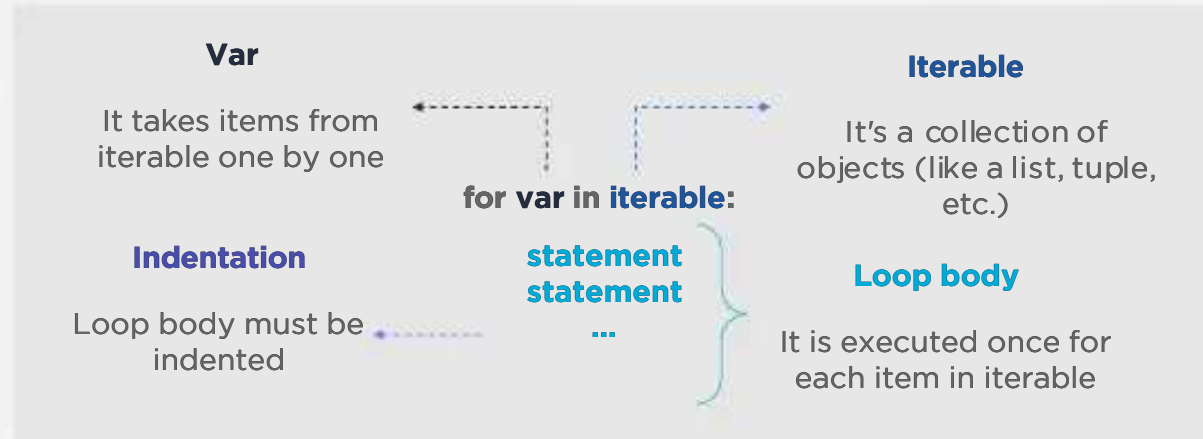
Ahora, entenderemos cómo utilizar la estructura de control repetitiva for.

Los temas que abordaremos son:

- Uso de la estructura de control For
- For anidado

# Uso de la estructura de control For

La estructura de control For genera un bucle que repite el bloque de instrucciones para un número predeterminado de veces. El bloque que se repite se llama cuerpo del bucle y cada repetición se denomina iteración.



# For anidado

Un For anidado es cuando dentro de un bucle For definimos 1 o más For.

## Ejemplo

Crear un programa que permita ingresar los nombres de N alumnos y la cantidad M de cursos por alumno.

```
3  N = int(input("Ingresar cantidad de alumnos: "))
4  for i in range(N):
5      nombre = input("ALUMNO {}: ".format(i))
6      M = int(input("Ingresar cantidad de cursos: "))
7      for j in range(M):
8          curso_nombre = input("CURSO {}: ".format(j+1))
```



**¿Por qué escribir 100 veces “Hola mundo” cuando lo puedes hacer con dos líneas utilizando For?**





# Estructura For: Break y Continue

Ahora, entenderemos cómo utilizar las estructuras de control repetitiva for utilizando break y continue.

Los temas que abordaremos son:

- Instrucción Break
- Instrucción Continue

# Instrucción Break

La instrucción Break permite terminar el bucle sin necesidad de realizar todas sus iteraciones.

```
1  for element in range(1, 100):  
2      print(element)  
3      break
```

```
1  i = 1  
2  
3  while i < 100:  
4      print(i)  
5      i += 1  
6      break
```

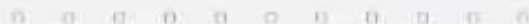
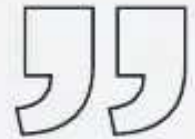
# Instrucción Continue

La instrucción Continue permite regresar a la condición del bucle.

```
1  for element in range(1, 10, 1):  
2      if element == 5:  
3          continue  
4      print(element)
```



**Todo fin es un nuevo comienzo**



## Ideas resumen del módulo 3

**01**

Para realizar acciones repetitivas dependientes de una condición se utiliza el While.

**02**

Si sabemos la cantidad de repeticiones se suele utilizar el For.

**03**

El break nos permite terminar con el bucle y el Continue regresa a consultar la condición del bucle.

# Bibliografía

- <https://keepcoding.io/blog/5-lenguajes-de-programacion-mas-usados-2022/>
- <https://duenaslerin.com/tico2/pdfs/python-para-todos.pdf>
- [https://bugs.python.org/file47781/Tutorial\\_EDIT.pdf](https://bugs.python.org/file47781/Tutorial_EDIT.pdf)

NETZUN