

Introducción a la programación

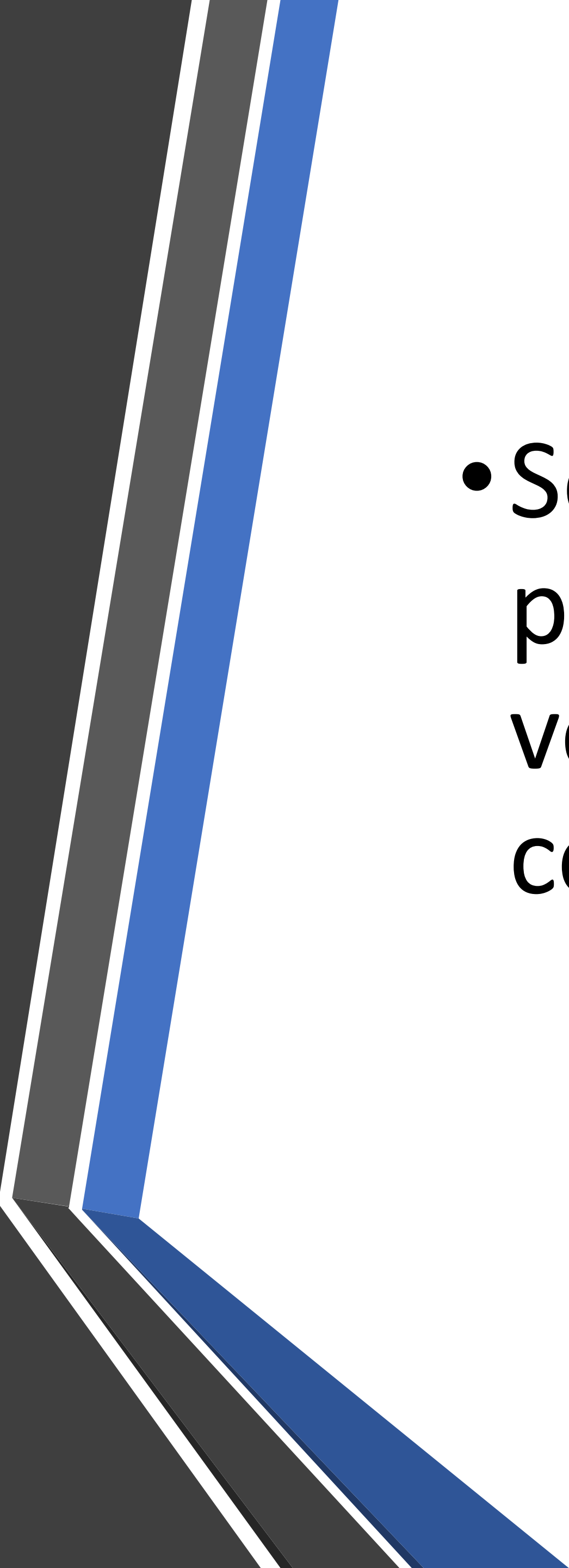
Profesor.: Lic. Jordani Moreira Briceño

Correo:
jordani.moreira.briceno@ucreativa.com

Agenda

- Tablas de Verdad
- Lógica de Proposiciones
 - ¿Qué es una proposición?
 - Negación
 - Conjunción
 - Disyunción
- Repaso de temas vistos utilizando Python.





¿Qué son las
tablas de
verdad?

- Son gráficos que nos sirven para conocer los valores de verdad de las proposiciones compuestas

¿De que se componen?

- Está compuesta de columnas y renglones

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

La proposición

La lógica Lenguaje preciso, claro y formal, que estudia a las proposiciones.

Su objetivo es interpretar, desarrollar razonamientos correctos y distinguirlos de los incorrectos.

- Proposición: Toda expresión lingüística que se afirma si es V o F

Ejemplo

P: Hoy es martes

Q: Los futbolistas son deportistas

R: Los perros vuelan

Ejemplo de Proposiciones

SON PROPOSICIONES

25 es divisible entre 3 .

$6 + 5 = 10$ ”.

El aula A1-205 está en el 2do piso

El sol es una estrella

Manuel saco 20 en matemática

Los problemas de matemática son fáciles

NO SON PROPOSICIONES

¡Pare inmediatamente!

En realidad, ¿a que se refiere?

Lávalo.

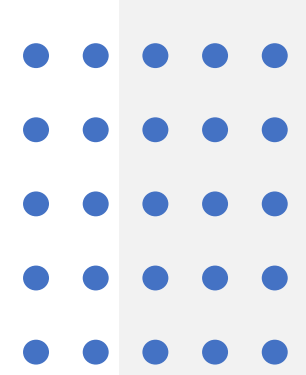
¡Que hermoso son tus ojos!

¿Lloverá mañana?

Haz esto por por favor.

Valor de la verdad

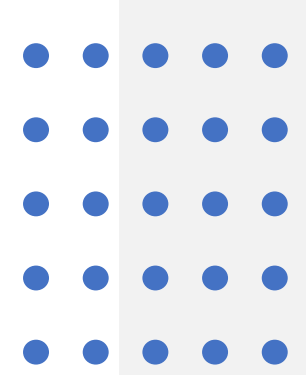
- Toda proposición se califica como verdadera (V) o falsa (F).
- **Ejemplos:**
 - La tierra es un satélite (F)
 - El conjunto unitario tiene un solo elemento (V)
 - 9 es cuadrado perfecto (V)
 - 3 mayor que 5 (F)



Conectivos
lógicos u
operadores
proposicionales:

Llamados también
operaciones lógicas.

Los mas importantes son: la
**negación, conjunción,
disyunción**



Conectivos lógicos u operadores proposicionales:

1. Negación (NOT)

- Pude usar las palabras como: **No, jamás, nunca.**

Ejemplos:

- El sol NO es un planeta.
- La tierra NO es un planeta del sistema solar

Matemáticamente se escribe así: $\sim p$

Negación (NOT)

- **Ejemplo**

- p : Nuestro salón está en el 2do piso.
- $\sim p$: Nuestro salón **no** está en el 2do piso.
- $\sim p$: **No es cierto que** nuestro salón esté en el 2do piso.
- Si p es verdadera entonces $\sim p$ es falsa. En cambio, si p es falsa, $\sim p$ es verdadera.

- **La tabla de verdad**

p	$\neg p$
V	F
F	V

La Conjunción ($p \wedge q$)

- La conjunción es verdadera, únicamente cuando ambas proposiciones que la componen son verdaderas.
- Si tuviera 3 variables sólo sería verdadero cuando las 3 fueran verdaderas.
- Ejemplo
 - Luis es médico y deportista.
 - El 4 es mayor que 2 y es un numero par.
- Valor de verdad: **$p \wedge q$ (variables)**
- Ejemplo:
- Juan tiene diez años también Elizabeth
- Benito perdió tanto dinero como Víctor.
- El sol sale aun cuando llueve.

Conjunción ($p \wedge q$) Inclusiva

- Ejemplos

- Tengo papel, pero no lápiz.
- Iremos a la playa si no llueve.
- María y Juan son novios.

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

La Disyunción $(p \vee q) / (p \underline{\vee} q)$

La disyuntiva es una alternativa entre dos enunciados puede ser **inclusiva** o **exclusiva**.

Regla: Es verdadera si al menos una de las dos es verdadera.

La Disyunción $(p \vee q) / (p \vee q)$

- **Inclusiva.**
- Siempre que utilicemos en el lenguaje natural la conjunción disyuntiva "o" en este sentido, utilizaremos el símbolo "**v**".
- Cuando decimos que para optar a un puesto de trabajo hay que saber inglés o francés, interpretamos que alguien que sabe inglés puede optar a dicho trabajo, alguien que sabe francés también, y, por supuesto, alguien que sepa tanto inglés o francés también.

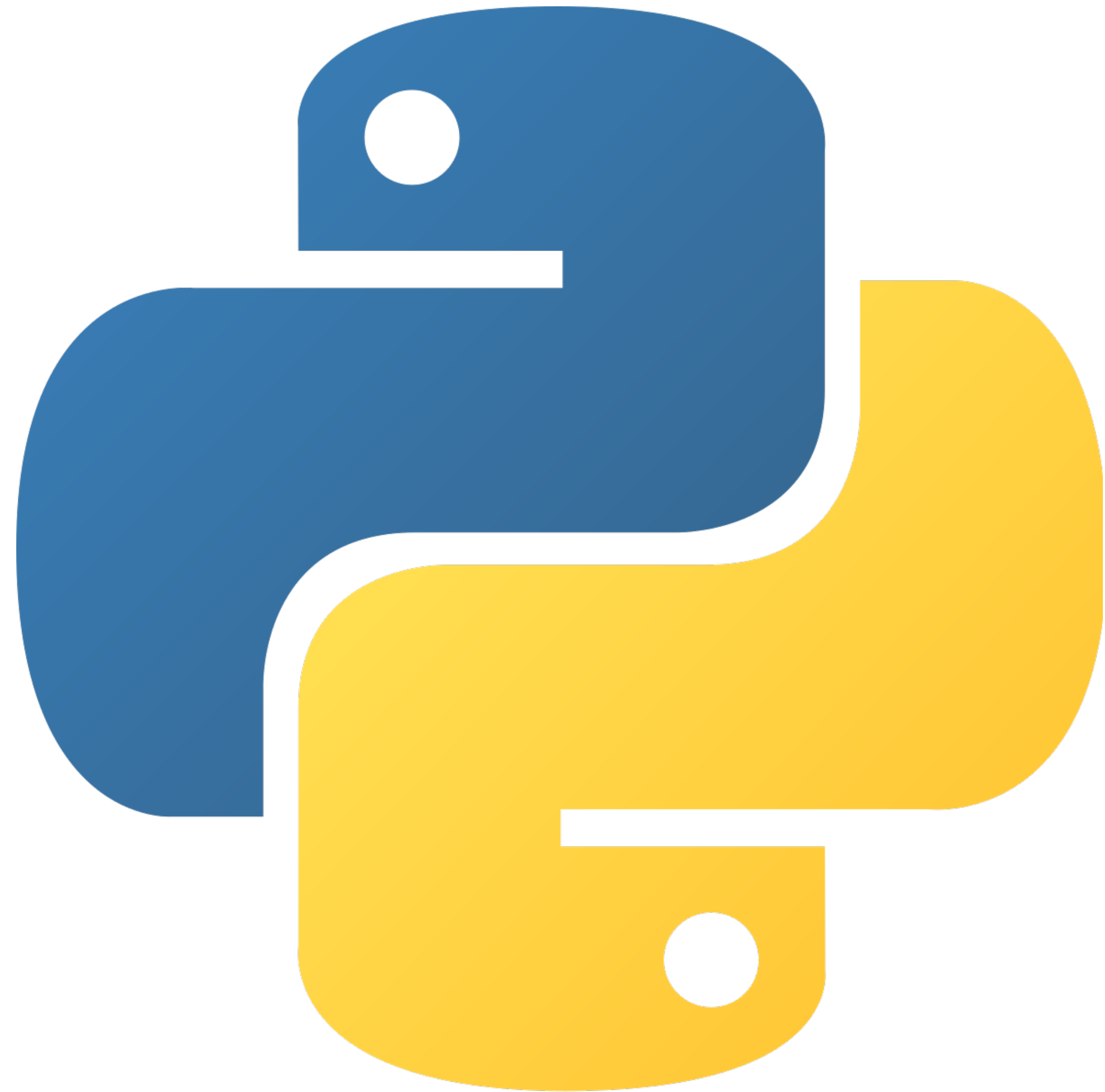
p	q	p v q
v	v	v
v	f	v
f	v	v
f	f	f

La Disyunción $(p \vee q) / (p \vee q)$

- **Exclusiva**
- En este sentido exclusivo si p es verdadera y q también lo es, la disyunción exclusiva es falsa.
- Por ejemplo, en el lenguaje natural empleamos este sentido exclusivo de la disyunción cuando decimos que alguien es cristiano o musulmán.
- Si alguien es cristiano, si es consecuente con ello no podrá ser musulmán, y viceversa.

p	q	$p \vee q$
v	v	f
v	f	v
f	v	v
f	f	f

Introducción
a Python Ver
3.9



¿Qué se necesita para programar en Python?

- ¿De donde lo obtengo?
- El interprete Python para Windows u otros sistemas operativos puede ser descargado de la pagina:
<https://www.python.org/downloads/>



Entorno de Desarrollo Python

TIPOS

Locales



Esta foto de Autor desconocido está bajo licencia [CC BY-NC](#)

Online

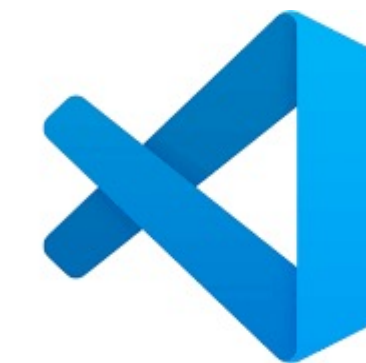
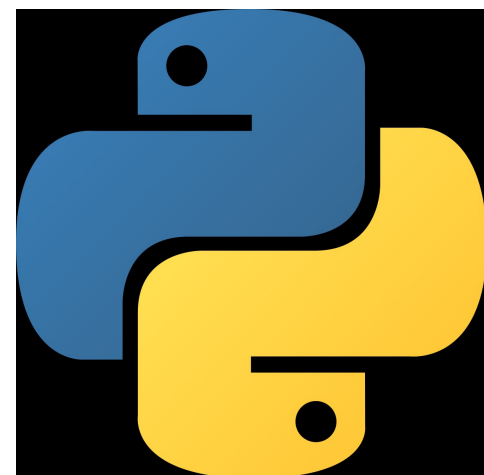


Portables



¿Qué se necesita para programar en Python?

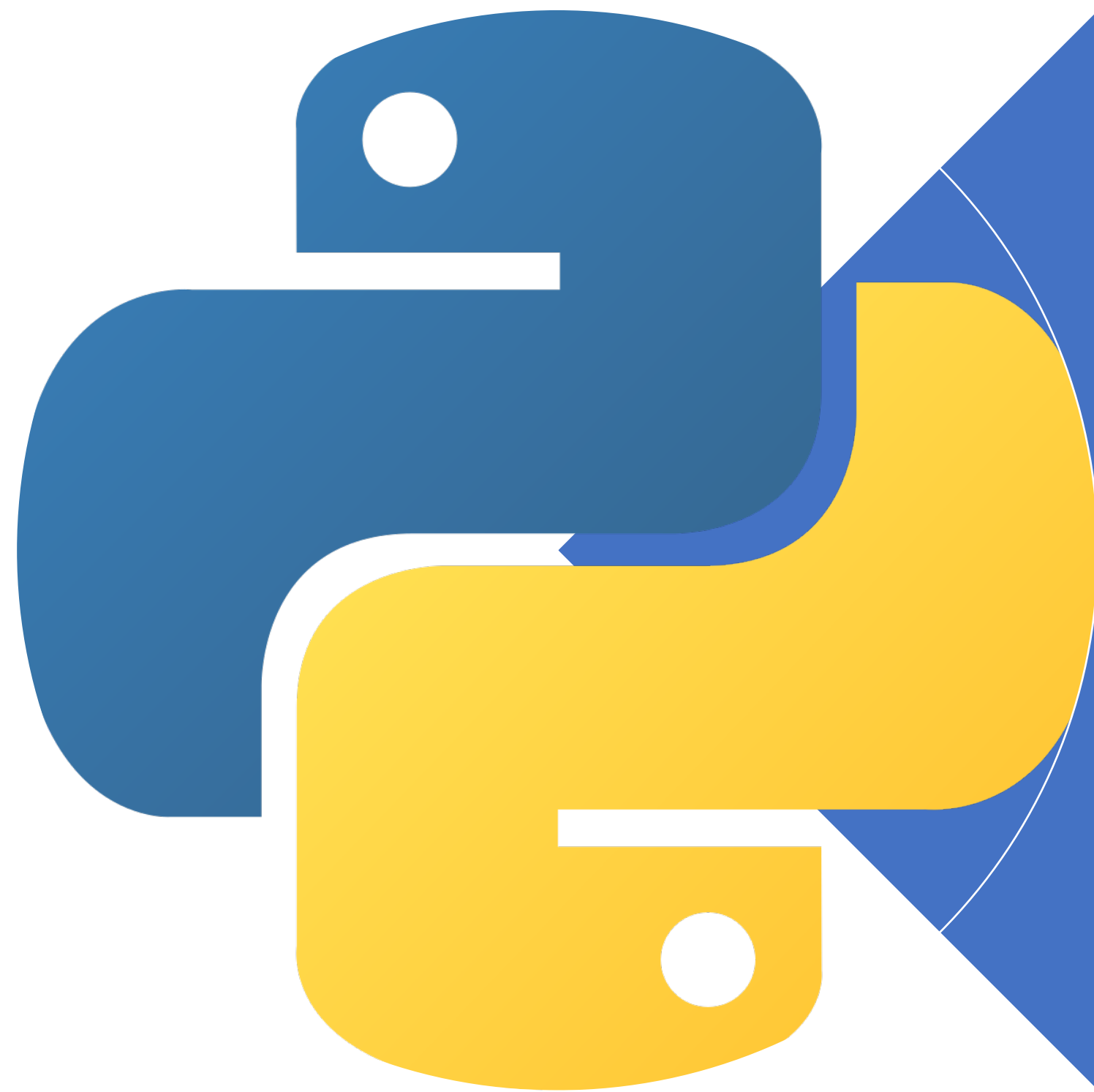
IDE y/o editores de texto	IDLE, Visual Studio Code, NetBeans,
Interprete / Compilador	Python 3.9
Sistema Operativos	Windows, Linux, MacOS
Hardware	Computadoras portátiles y de escritorio.



¿Cuál versión instalar?

- Es posible tener instalados en el ordenador varias versiones de Python, pero salvo que sea necesario para la ejecución de programas o paquetes incompatibles se recomienda instalar siempre la última versión disponible.
- La transición de Python 2 a Python 3 está resultando mucho más costosa de lo esperado, seguramente porque Python 3 introdujo muchos cambios en el lenguaje y obliga a reescribir prácticamente todos los programas (aunque se han creado herramientas para ayudar en ese proceso)

Python



Programa estructurado

- Todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:
 - Secuencial.
 - Instruccional condicional.
 - Iteración (bucle de instrucciones).

Programa en python

- Un programa Python está compuesto por una secuencia de instrucciones que son ejecutadas por el intérprete en una terminal.
 - `print('Hola')`
 - `can_persona = 12`
 - `print('Hay ', can_persona, ' personas')`
-

Tipos de datos (**Recordatorio**)

Número Entero (int)

- Este tipo de dato se corresponde con números enteros, es decir, sin parte decimal.

Número Decimal (float)

- Este tipo de dato se corresponde con números reales con parte decimal. Cabe destacar que el separador decimal en Python es el punto (.) y no la coma (,).

Caracter (chr)

- Este tipo de dato se corresponde con un símbolo tipográfico, es decir, una letra, número, coma, espacio, signo de puntuación, etc.

Cadena de Texto (str)

- Este tipo de datos se corresponde con una cadena de caracteres.

Booleano (bool)

- Este tipo de dato reconoce solamente dos valores: Verdadero (True) y Falso (False)

Operadores Aritméticos (**Recordatorio**)

OPERADOR	DESCRIPCIÓN	USO
+	Realiza Adición entre los operandos	$12 + 3 = 15$
-	Realiza Substracción entre los operandos	$12 - 3 = 9$
*	Realiza Multiplicación entre los operandos	$12 * 3 = 36$
/	Realiza División entre los operandos	$12 / 3 = 4$
%	Realiza un módulo entre los operandos	$16 \% 3 = 1$
**	Realiza la potencia de los operandos	$12 ** 3 = 1728$
//	Realiza la división con resultado de número entero	$18 // 5 = 3$

Operadores Relacionales o de comparación

Un operador relacional se emplea para comparar y establecer la relación entre ellos. Devuelve un valor booleano (true o false) basado en la condición.

OPERADOR	DESCRIPCIÓN	USO
>	Devuelve True si el operador de la izquierda es mayor que el operador de la derecha	12 > 3 devuelve True
<	Devuelve True si el operador de la derecha es mayor que el operador de la izquierda	12 < 3 devuelve False
==	Devuelve True si ambos operandos son iguales	12 == 3 devuelve False
>=	Devuelve True si el operador de la izquierda es mayor o igual que el operador de la derecha	12 >= 3 devuelve True
<=	Devuelve True si el operador de la derecha es mayor o igual que el operador de la izquierda	12 <= 3 devuelve False
!=	Devuelve True si ambos operandos no son iguales	

Operadores de Pertenencia

- Un operador de pertenencia se emplea para identificar pertenencia en alguna secuencia (listas, strings, tuplas).
- **in** y **not in** son operadores de pertenencia.
- **in** devuelve True si el valor especificado se encuentra en la secuencia. En caso contrario devuelve False.
- **not in** devuelve True si el valor especificado no se encuentra en la secuencia. En caso contrario devuelve False.

```
a = [1,2,3,4,5]
```

```
#Esta 3 en la lista a?
```

```
print 3 in a # Muestra True
```

```
#No está 12 en la lista a?
```

```
print 12 not in a # Muestra True
```

```
str = "Hello World"
```

```
#Contiene World el string str?
```

```
print "World" in str # Muestra True
```

```
#Contiene world el string str? (nota: distingue mayúsculas y minúsculas)
```

```
print "world" in str # Muestra False
```

```
print "code" not in str # Muestra True
```

Syntaxis

Correcta

- `if 5 > 2:`
 `print("Five is greater`
 `than two!")`

Incorrecta

- `if 5 > 2:`
 `print("Five is greater`
 `than two!")`

Comentarios

Los comentarios se pueden utilizar para explicar el código Python.

Los comentarios se pueden utilizar para hacer que el código sea más legible.

Los comentarios se pueden utilizar para evitar la ejecución al probar el código.

Comentarios

Los comentarios comienzan con a **#**, y Python los ignorará:

```
#Esto es un comentario  
print("Hola Mundo.")
```

Los comentarios se pueden colocar al final de una línea y Python ignorará el resto de la línea:

```
print("Hola Mundo") #Esto es un comentario
```

Comentario de varias líneas

```
#Esto es un comentario  
#se pueden escribir  
#más líneas como necesite  
print("Hello, World!")
```

Comentarios

- Dado que Python ignorará los literales de cadena que no están asignados a una variable, puede agregar una cadena de varias líneas (comillas triples) en su código y colocar su comentario dentro de ella:

```
"""
```

```
This is a comment  
written in  
more than just one line  
"""
```

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

Lectura

Para capturar datos que el usuario quiera ingresar al programa...

```
name = input('Cual es su nombre?')  
print('Buenas noches', name, '!')  
print('Buenas noches' + name + '!')
```

Práctica/Tarea

Utilice los diagramas de flujo de la asignación número 1 y transcribalos a código python.

