



FUNDAMENTOS DE PYTHON

HIGH LEVEL LANGUAGE

BLOQUE DE APRENDIZAJE
AUTONOMO GENERALIZADO

2024





Bloque de Aprendizaje Autónomo Generalizado

Guía de Manejo del lenguaje

Gabriel Agustín Miño Villalobos



2024



INDICE

- 1. Introducción al Lenguaje
- 2. Variables
- 3. Operadores
- 4. Control de Cadenas
- 5. Flujo de Condicionales
- 6. Bucles
- 7. Manejo de Listas y Tuplas
- 8. Funciones
- 9. Generadores
- 10. Diccionarios
- 11. Programación Funcional
- 12. Ficheros
- 13. Depuración
- 14. Manejo de Librerías
- 15. Excepciones
- 16. Introducción a POO
- 17. Constructores
- 18. Encapsulamiento
- 19. Herencia
- 20. Strings
- 21. Módulos y Paquetes
- 22. Archivos de Texto
- 23. Archivos Binarios
- 24. Ventanas
- 25. Controlador / Variables GUI
- 26. Texto Multilínea y Botones
- 27. Menús
- 28. Explorador y Listas

REPOTEADO POR:



CAPITULO 1: Introducción al Lenguaje

PROGRAMACION

SE BASA EN:

El arte de escribir instrucciones para que las computadoras realicen tareas específicas. Se basa en conceptos como variables, estructuras de control y funciones.

La escritura de código para instruir a una computadora sobre cómo llevar a cabo ciertas tareas o resolver problemas específicos.



Hoy en día, la programación es una habilidad fundamental en una amplia variedad de campos, desde el desarrollo de software hasta la ciencia de datos, la inteligencia artificial y más allá. Se ha convertido en una herramienta poderosa para resolver problemas y crear tecnología innovadora que impulsa el mundo moderno.

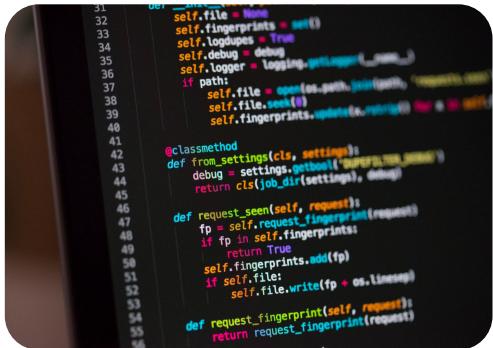
La programación es una habilidad esencial en el mundo moderno, con aplicaciones que abarcan desde el desarrollo de software hasta la ciencia de datos y la inteligencia artificial. Permite crear aplicaciones y programas de software, analizar grandes volúmenes de datos, desarrollar sistemas de inteligencia artificial, diseñar sitios web dinámicos, automatizar tareas repetitivas, garantizar la seguridad informática y crear experiencias de entretenimiento.



La programación puede realizarse en varios lenguajes de programación, cada uno con sus propias reglas sintácticas y semánticas. Estos lenguajes permiten a los programadores expresar sus instrucciones de manera que la computadora pueda entenderlas y ejecutarlas.

LENGUAJES DE PROGRAMACIÓN

PYTHON



A screenshot of a terminal window displaying Python code. The code is a class definition with various methods and attributes, including file handling and fingerprinting logic. The code is color-coded for syntax highlighting.

```
31     def __init__(self, file=None, fingerprints=None):
32         self.file = file
33         self.fingerprints = fingerprints
34         self.logupes = True
35         self.debug = debug
36         self.logger = logging.getLogger(__name__)
37         if path:
38             self.file = open(os.path.join(path, 'fingerprint.log'), 'w')
39             self.file.seek(0)
40             self.fingerprints.update(self.file)
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getboolean('FINGERPRINT_DEBUG')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54         self.request_fingerprint(self, request)
55
56     def request_fingerprint(self, request):
57         return request_fingerprint(request)
```

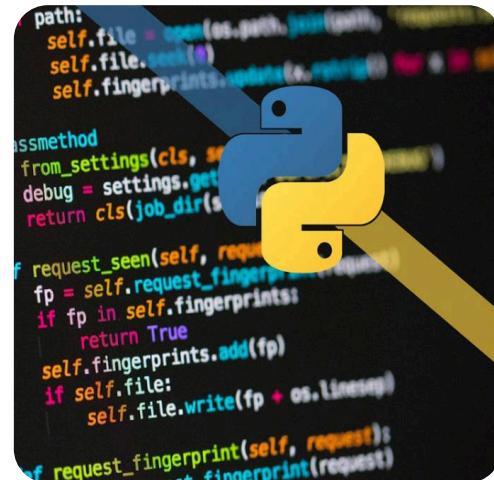
SE BASA EN:

Un lenguaje de programación de alto nivel conocido por su simplicidad y legibilidad. Es versátil y puede ser utilizado en una amplia gama de aplicaciones, desde desarrollo web hasta inteligencia artificial. Python cuenta con una gran cantidad de bibliotecas y frameworks que facilitan el desarrollo de proyectos.

- Python es una excelente opción para aquellos que están empezando en la programación debido a su simplicidad, versatilidad y gran comunidad de soporte. Además, su amplio uso en una variedad de campos lo convierte en una habilidad valiosa para aprender.
- Python tiene una sintaxis clara y simple que facilita la lectura y escritura de código. Su diseño enfocado en la legibilidad.
- Python para desarrollar desde simples scripts hasta aplicaciones web complejas, análisis de datos, inteligencia artificial.
- Python cuenta con una gran cantidad de bibliotecas y frameworks que facilitan el desarrollo de una variedad de proyectos. Por ejemplo, Django y Flask son frameworks populares para el desarrollo web, mientras que NumPy y pandas son bibliotecas populares para la manipulación y análisis de datos.
- Python es altamente valorado en el campo de la inteligencia artificial y el machine learning, siendo una herramienta clave para el desarrollo de algoritmos y modelos predictivos. Su facilidad para implementar conceptos complejos de forma sencilla lo convierte en una elección popular entre los científicos de datos y desarrolladores.

El lenguaje de programación Python es un lenguaje multiparadigma y multiplataforma.

Que sea multiparadigma significa que con este lenguaje se pueden construir programas con enfoques diferentes, a saber: enfoque funcional (se privilegia el procesamiento), enfoque imperativo (se privilegia el almacenamiento en la memoria principal) y enfoque orientado a objetos (se privilegia la unión entre atributos, que son las características de las cosas u objetos, y los métodos, que son los usos que pueden tener dichas cosas u objetos).



Un lenguaje multiparadigma brinda a los desarrolladores la capacidad de adaptarse fácilmente a estos diferentes contextos, lo que les permite escribir código más eficiente y efectivo.

Al ser multiparadigma, Python puede adaptarse a una variedad de estilos de programación, como programación orientada a objetos, programación funcional, programación imperativa, etc.

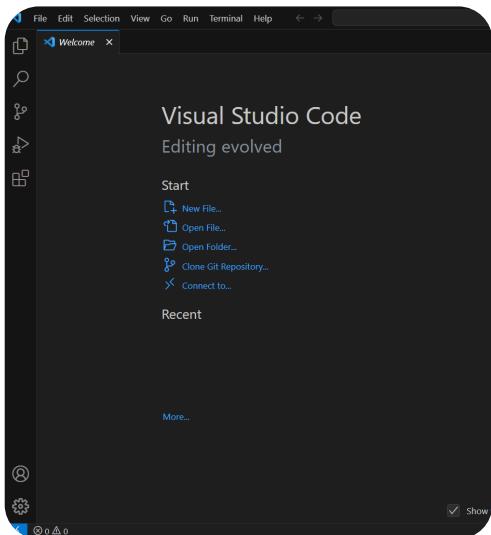
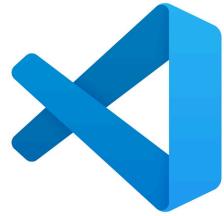


Esto permite a los desarrolladores elegir el enfoque más adecuado para cada situación.

Facilita la integración de código existente en un proyecto, independientemente del estilo de programación utilizado para escribirlo originalmente.

COMPONENTES

VISUAL STUDIO CODE



SE BASA EN:

Un editor de código fuente desarrollado por Microsoft. Es un entorno de desarrollo ligero, gratuito y de código abierto que es altamente personalizable y está diseñado para funcionar en múltiples plataformas. Admite una amplia variedad de extensiones desarrolladas por la comunidad, lo que permite a los usuarios personalizar y ampliar su funcionalidad para adaptarse a sus necesidades específicas de desarrollo.

EXTENSIONES

CODESNAP



SE BASA EN:

Una característica específica en Visual Studio Code, que es un popular editor de código fuente desarrollado por Microsoft. Esta función permite a los usuarios tomar capturas de pantalla del código que están editando en el editor y compartir las fácilmente con otros desarrolladores o colaboradores.



CODEIUM



Autocomplete | Chat | Search

Codeium: Refactor | Explain | Generate Docstring

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import pandas as pd
4
5 def plot_two_variables(data, col_a, col_b):
6     """
7         Plots two variables from the given data
8
9     Args:
10        data (pandas.DataFrame): The data c
11        col_a (str): The name of the column
12        col_b (str): The name of the column
13
14    Returns:
15        None
16    """
17    plt.scatter(data[col_a], data[col_b])
18    m, b = np.polyfit(data[col_a], data[col_b], 1)
19    plt.plot(data[col_a], m * data[col_a] + b)
20    plt.xlabel(col_a)
21    plt.ylabel(col_b)
22    plt.show()
```

SE BASA EN:

Una herramienta de aceleración de código gratuito creado con tecnología de inteligencia artificial de vanguardia. Actualmente, Codeium proporciona una herramienta para completar código en más de 70 lenguajes, con velocidades ultrarrápidas y una calidad de sugerencias de última generación. Acelera el proceso de iteración de su software, mejorar la calidad y la coherencia de su código, reducir la cantidad de iteraciones de revisión de código, acelerar la incorporación de desarrolladores y mantenerlos en su estado de flujo.

VERSION LENS



SE BASA EN:

Proporcionar información sobre las dependencias de los proyectos gestionados por sistemas de control de versiones como npm (Node Package Manager), Maven (para proyectos Java), y otros. Esta herramienta útil para los desarrolladores que trabajan con proyectos que tienen dependencias gestionadas por sistemas de control de versiones, ya que les ayuda a mantenerse al tanto de las últimas versiones disponibles de las dependencias y a tomar decisiones informadas sobre cuándo actualizarlas.

```
"dependencies": {
    "satisfies": "0.26.1 | latest: ↑ 0.27.2",
    "axios": "^0.26.1",
    "satisfies": "latest | latest: ↑ 2.29.3",
    "moment": "^2.29.2",
    "fixed": "12.1.4 | latest: ↑ 12.1.6",
    "next": "12.1.4",
    "satisfies": "latest | latest: ↑ 5.5.2",
    "next-pwa": "^5.5.0",
    "fixed": "18.0.0 | latest: ↑ 18.1.0",
    "react": "18.0.0",
    "fixed": "18.0.0 | latest: ↑ 18.1.0",
    "react-dom": "18.0.0"
},
```

PYTHON EXTENSION PACK



SE BASA EN:

Un conjunto de extensiones para Visual Studio Code (VS Code) que están específicamente diseñadas para mejorar la experiencia de desarrollo de Python en este entorno. Estas extensiones están empaquetadas juntas para facilitar su instalación y configuración.

PYTHON INDENT



SE BASA EN:

La indentación, o sangría, en Python sirve como una parte fundamental de la estructura y sintaxis del código en este lenguaje de programación. A diferencia de otros lenguajes de programación que utilizan llaves {} u otros delimitadores para definir bloques de código, Python utiliza la indentación para delimitar bloques de código, como bucles, funciones, clases y estructuras de control.

```
name = 'Sam'  
if name == 'Sam':  
    print('Hello Sam')  
else:  
    print('Hello dude')  
    print('How are you?')
```

Without Indentation

```
name = 'Sam'  
if name == 'Sam':  
    print('Hello Sam')  
else:  
    print('Hello dude')  
print('How are you?')
```

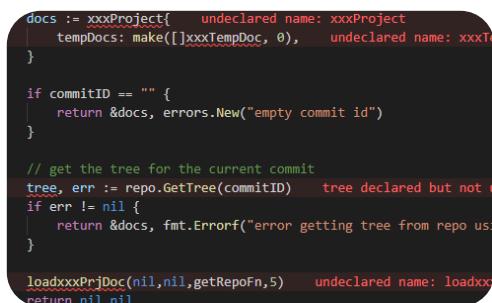
With Indentation

ERROR LENS



SE BASA EN:

Mostrar los errores de tu código en la misma línea en lugar de aparecer en el status bar. Mejora tu productividad detectando más rápidamente problemas sin necesidad de hacer hover en tu código.

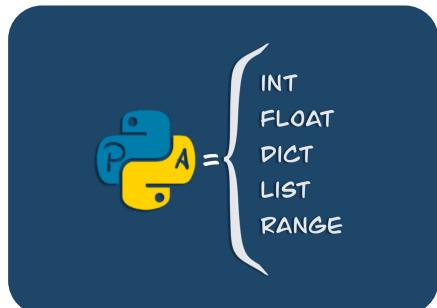


CAPITULO 2: Variables

VARIABLES

SE BASA EN:

Un espacio de memoria en donde se puede guardar un dato. En una variable se puede guardar un número (como 4 o 3,8) o una cadena de caracteres (como "Omar" o "UTP").



TIPOS DE DATOS

- **Enteros (int):** Números enteros, como 5, -3, 1000, etc.



```
1 numero_entero=int(input("introduce un numero entero: "))
```

- **Flotantes (float):** Números de punto flotante, que representan números decimales, como 3.14, -0.001, etc.



```
1 numero_flotante=float(input("introduce un numero flotante: "))
```

- **Booleanos (bool):** Valores booleanos que pueden ser Verdadero (True) o Falso (False).



- **Cadenas de caracteres (str):** Secuencias de caracteres, como "Hola mundo", 'Python', etc.

```
● ● ●
1 objeto=str(input("introduce un objeto: "))
2 print (objeto)
```

- **Listas (list):** Colecciones ordenadas y modificables de elementos, que pueden ser de diferentes tipos.

```
● ● ●
1 # Creación de una lista de números enteros
2 lista_enteros = [1, 2, 3, 4, 5]
3
4 # Creación de una lista de cadenas de caracteres
5 lista_cadenas = ["hola", "mundo", "Python"]
6
7 # Creación de una lista mixta
8 lista_mixta = [1, "dos", True, 3.14]
9
10 # Creación de una lista vacía
11 lista_vacia = []
12
13 # Acceso a elementos de la lista
14 print(lista_enteros[0]) # Imprime el primer elemento de la lista_enteros (1)
15 print(lista_cadenas[2]) # Imprime el tercer elemento de la lista_cadenas ("Python")
```

- **Tuplas (tuple):** Colecciones ordenadas e inmutables de elementos.

```
● ● ●
1 # Creación de una tupla de números enteros
2 mi_tupla = (1, 2, 3, 4, 5)
3
4 # Creación de una tupla de cadenas de caracteres
5 tupla_cadenas = ("hola", "mundo", "Python")
6
7 # Creación de una tupla mixta
8 tupla_mixta = (1, "dos", True, 3.14)
9
10 # Acceso a elementos de la tupla
11 print(mi_tupla[0]) # Imprime el primer elemento de la tupla (1)
12 print(tupla_cadenas[2]) # Imprime el tercer elemento de la tupla_cadenas ("Python")
13
14 # Intentando modificar un elemento de la tupla (esto generará un error)
15 # mi_tupla[0] = 100 # Esto generará un TypeError: 'tuple' object does not support item assignment
16
```

DIFERENCIAS ENTRE TUPLAS Y LISTAS

- **Tupla:**

- Una tupla es una secuencia inmutable de elementos.
- Se define utilizando paréntesis ().
- Los elementos de una tupla no pueden ser modificados una vez que la tupla ha sido creada.
- Las tuplas son más eficientes en términos de espacio y rendimiento que las listas, ya que ocupan menos memoria y son más rápidas de iterar.
- Son adecuadas para almacenar colecciones de elementos que no cambiarán durante la ejecución del programa, como coordenadas geográficas, información de un registro de base de datos, etc.

- **Lista:**

- Una lista es una secuencia mutable de elementos.
- Se define utilizando corchetes [].
- Los elementos de una lista pueden ser modificados, añadidos o eliminados después de que la lista ha sido creada.
- Las listas son menos eficientes en términos de espacio y rendimiento que las tuplas, ya que ocupan más memoria y son más lentas de iterar.
- Son adecuadas para almacenar colecciones de elementos que pueden cambiar dinámicamente durante la ejecución del programa, como una lista de tareas pendientes, nombres de estudiantes en una clase, etc.

CHULETA	Tupla	Lista	Diccionarios
Definición	<code>mi_tupla = ('texto', 20, 1275.48)</code>	<code>mi_lista = ['texto', 20, 1275.48]</code>	<code>mi_dict = {'clave uno':'texto', 'clave dos':20, 'clave tres':1275.48}</code>
Obtener uno de los valores	<code>print mi_tupla[0] # imprime texto print mi_tupla[1] # imprime 20 print mi_tupla[2] # imprime 1275.48</code>	<code>print mi_lista[0] # imprime texto print mi_lista[1] # imprime 20 print mi_lista[2] # imprime 1275.48</code>	<code>print mi_dict['clave uno'] # imprime texto print mi_dict['clave dos'] # imprime 20 print mi_dict['clave tres'] # imprime 1275.48</code>
Modificar uno de sus valores	NO SE PUEDE	<code>mi_lista[0] = 'cambió'</code>	<code>mi_dict['clave dos'] = 34</code>

- **Diccionarios (dict):** Una variable de tipo diccionario en Python es una estructura de datos que permite almacenar una colección de pares clave-valor, donde cada clave está asociada a un valor específico. En un diccionario, las claves deben ser únicas e inmutables, mientras que los valores pueden ser de cualquier tipo de dato, incluso otros diccionarios.

```

● ● ●
1 # Creación de un diccionario de información de una persona
2 persona = {
3     "nombre": "Juan",
4     "apellido": "Pérez",
5     "edad": 30,
6     "ciudad": "Ciudad de México"
7 }
8
9 # Acceso a elementos del diccionario
10 print(persona["nombre"]) # Imprime el valor asociado a la clave "nombre" ("Juan")
11 print(persona["edad"]) # Imprime el valor asociado a la clave "edad" (30)
12
13 # Modificación de elementos del diccionario
14 persona["edad"] = 35 # Modifica el valor asociado a la clave "edad" a 35
15 print(persona["edad"]) # Imprime el nuevo valor asociado a la clave "edad" (35)
16
17 # Agregar un nuevo elemento al diccionario
18 persona["profesion"] = "Ingeniero"
19 print(persona) # Imprime el diccionario completo, incluyendo el nuevo elemento
20

```

- **Conjuntos (set):** Las colecciones permiten almacenar datos de diferentes tipos y estructuras en un solo contenedor. Esto facilita el manejo de grandes cantidades de datos en un programa.

```

● ● ●
1 # Ejemplo de colecciones en Python
2
3 # Lista de nombres
4 nombres = ["Juan", "María", "Pedro", "Ana"]
5
6 # Tupla de edades
7 edades = (30, 25, 35, 28)
8
9 # Conjunto de ciudades
10 ciudades = {"Ciudad de México", "Madrid", "Lima", "Buenos Aires"}
11
12 # Diccionario de información personal
13 info_personal = {
14     "Juan": {"edad": 30, "ciudad": "Ciudad de México"},
15     "María": {"edad": 25, "ciudad": "Madrid"},
16     "Pedro": {"edad": 35, "ciudad": "Lima"},
17     "Ana": {"edad": 28, "ciudad": "Buenos Aires"}
18 }
19
20 # Acceder a elementos de las colecciones
21 print("Primer nombre en la lista:", nombres[0])
22 print("Edad de María:", edades[1])
23 print("Ciudades disponibles:", ciudades)
24
25 # Acceder a información personal de Juan
26 print("Información personal de Juan:", info_personal["Juan"])

```

CAPITULO 3: Operadores

OPERADORES

SE BASA EN:

Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.

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	$18 // 5 = 3$

TIPOS DE OPERADORES

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

```
● ● ●
1 a=10 #variable de tipo entero
2 b=2 #variable de tipo entero
3 suma=a+b
4 resta=a-b
5 multiplicacion=a*b
6 division=a/b
7 cociente=a//b
8 residuo=a%b
9 potencia=a**b
10
11 print('este es el valor de la suma',suma) #resultado es 12
12 print('este es el valor de la resta',resta) #resultado es 8
13 print('este es el valor de la multiplicacion',multiplicacion )#resultado es 20
14 print('este es el valor de la division ',division) #resultado es 5
15 print('este es el valor de la cociente',cociente) #resultado es 5
16 print('este es el valor de la residuo',residuo) #resultado es 0
17 print('este es el valor de la potencia ',potencia) #resultado es 100
```

Operadores de comparación: Se utilizan para comparar valores y devolver un valor booleano Verdadero (True) o Falso (False). Algunos ejemplos incluyen igualdad (`==`), desigualdad (`!=`), mayor que (`>`), menor que (`<`), mayor o igual que (`>=`), y menor o igual que (`<=`).

```
● ○ ●
1 # Definir dos variables
2 a = 10
3 b = 20
4
5 # Ejemplos de operadores de comparación
6 print("a es igual a b:", a == b)
7 print("a es diferente de b:", a != b)
8 print("a es mayor que b:", a > b)
9 print("a es menor que b:", a < b)
10 print("a es mayor o igual que b:", a >= b)
11 print("a es menor o igual que b:", a <= b)
12
13 # False, a no es igual a b
14 # True, a es diferente de b
15 # False, a no es mayor que b
16 # True, a es menor que b
17 # False, a no es mayor o igual que b
18 # True, a es menor o igual que b
```

Operadores lógicos: Se utilizan para combinar expresiones booleanas y devolver un resultado booleano. Los operadores lógicos comunes incluyen "y" (and), "o" (or) y "no" (not).

```
● ○ ●
1 # Definir tres variables
2 x = 5
3 y = 10
4 z = 15
5
6 # Ejemplos de operadores lógicos
7
8 # Operador "and":
9 print("x < y and y < z:", x < y and y < z)
10 # Devuelve True si ambas condiciones son verdaderas
11 # True, x es menor que y y es menor que z
12
13 # Operador "or":
14 print("x < y or y > z:", x < y or y > z)
15 # Devuelve True si al menos una de las condiciones es verdadera
16 # True, x es menor que y o y es mayor que z
17
18 # Operador "not":
19 print("not(x < y):", not(x < y))
20 # Devuelve el inverso del valor booleano
21 # False, x es menor que y, pero not invierte el resultado a False
```

Operadores de asignación: Se utilizan para asignar valores a variables. El operador de asignación básico es `"=`, pero también hay operadores de asignación compuestos como `"+="`, `"-="`, `"*="`, etc., que realizan una operación y luego asignan el resultado a la variable.

```
● ○ ●
1 # Definir una variable
2 x = 5
3 print("Valor inicial de x:", x)
4
5 # Ejemplos de operadores de asignación
6 x += 3 # Equivalente a: x = x + 3
7
8 x -= 2 # Equivalente a: x = x - 2
9
10 x *= 4 # Equivalente a: x = x * 4
11
12 x /= 2 # Equivalente a: x = x / 2
13
14 x %= 3 # Equivalente a: x = x % 3
15
16 x **= 2 # Equivalente a: x = x ** 2
```

EJERCICIOS DE PRACTICA

BLOQUE I

- Construir un programa que reciba dos números enteros y calcule el resultado de elevar el 1º número leído a la potencia representada por el 2º número leído

Solución #1



```
1 num1=int(input('enter a number'))
2 numb2=int(input('enter a number'))
3 result=num1**numb2
4 print('the result is',result)
```

- Construir un programa que reciba un número entero y obtenga su mitad entera

Solución #1



```
1 num1=int(input('enter a number'))
2 result=num1/2
3 print('the result is',result)
```

- Construir un programa que reciba un número entero y muestre su último dígito

Solución #1



```
1 num1=int(input('enter a number: '))
2 result=num1%10
3 print('the last digit is',result)
```

- Construir un programa que reciba un número entero y muestre el resultado de sumar sus dos últimos dígitos.

Solución #1

```
● ● ●  
1 num1=(input('enter a number: '))  
2 num1=num1[::-1]  
3 sum=int(num1[0])+int(num1[1])  
4 #Es necesario convertir el string a entero  
5 print('the addition is',sum)
```

- Construir un programa que reciba un número entero y muestre el resultado de elevar su penúltimo dígito a la potencia representada por su último dígito.

Solución #1

```
● ● ●  
1 num1=(input('enter a number: '))  
2 num1=num1[::-1]  
3 pot=int(num1[1])**int(num1[0])  
4 #Es necesario convertir el string a entero  
5 print('the addition is',pot)
```

- Construir un programa que reciba un número entero y muestre el residuo de dividirlo entre 6 y entre 8.

Solución #1

```
● ● ●  
1 num1=int(input('enter a number: '))  
2 res6=num1%6  
3 res8=num1%8  
4 print('the residual of 6 is',res6)  
5 print('the residual of 8 is',res8)
```

- Construir un programa que reciba dos números enteros y muestre el resultado de sumar el último dígito de cada número.

Solución #1

```
● ● ●  
1 num1=(input('enter a number: '))  
2 num2=(input('enter another number: '))  
3 num1=num1[::-1]  
4 num2=num2[::-1]  
5 sum=int(num1[0])+int(num2[0])  
6 print('the sum is: ',sum)
```

- Construir un programa que reciba dos números y muestre la suma de sus tres últimos dígitos

Solución #1

```
● ● ●  
1 num1=(input('enter a number: '))  
2 num2=(input('enter another number: '))  
3 num1=num1[::-1]  
4 num2=num2[::-1]  
5 sum1=int(num1[0])+int(num1[1])+int(num1[2])  
6 sum2=int(num2[0])+int(num2[1])+int(num2[2])  
7 print('the sum is: ',sum1+sum2)
```

- Construir un programa que reciba dos cadenas y determine si las dos cadenas son exactamente iguales.

Solución #1

```
● ● ●  
1 num1=(input('enter a number: '))  
2 num2=(input('enter another number: '))  
3 print('Those numbers are equal?: ', num1 == num2)
```

BLOQUE II

- Escribir un programa que muestre por pantalla el resultado de la siguiente operación aritmética $(3+2 \cdot 5)^2$.

Solución #1

```
● ● ●  
1 print(((3 + 2) / (2 * 5)) ** 2)  
2 #resultado: 0.25
```

Solución #2

```
● ● ●  
1 a=3+2  
2 b=2*5  
3 c=a/b  
4 d=c**2  
5 print("el resultado es", d)  
6 # resultado es 0.25
```

- Escribir un programa que pregunte al usuario por el número de horas trabajadas y el coste por hora. Después debe mostrar por pantalla la paga que le corresponde.

Solución #1

```
● ● ●  
1 horas = float(input("Introduce tus horas de trabajo: "))  
2 coste = float(input("Introduce lo que cobras por hora: "))  
3 paga = horas * coste  
4 print("Tu paga es", paga)
```

Solución #2

```
● ● ●  
1 a=int(input("introduce un numero de horas trabajadas "))  
2 b=int(input("introduce el coste por hora "))  
3 print("el salario total es", (a*b))
```

- Escribir un programa que pida al usuario su peso (en kg) y estatura (en metros), calcule el índice de masa corporal y lo almacene en una variable, y muestre por pantalla la frase Tu índice de masa corporal es <imc> donde <imc> es el índice de masa corporal calculado redondeado con dos decimales.

Solución #1

```
● ● ●  
1 p=float(input("cuál es tu peso en kg? "))  
2 a=float(input("cuál es tu altura en m? "))  
3 imc=round(p/a**2,2) #p/(a**2)  
4 print("tu indice de masa corporal es",imc)
```

Solución #2

```
● ● ●  
1 peso = input("¿Cuál es tu peso en kg? ")  
2 estatura = input("¿Cuál es tu estatura en metros? ")  
3 imc = round(float(peso)/float(estatura)**2,2)  
4 print("Tu indice de masa corporal es " + str(imc))
```

- Escribir un programa que lea un entero positivo, n , introducido por el usuario y después muestre en pantalla la suma de todos los enteros desde 1 hasta n . La suma de los n primeros enteros positivos puede ser calculada de la siguiente forma:

$$\text{suma} = n(n+1)/2$$

Solución #1



```
1 n=int(input("introduce un numero entero: "))
2 print("la suma desde 1 hasta",n,"es",((n*(n+1))/2))
```

Solución #2



```
1 n = int(input("Introduce un número entero: "))
2 suma = n * (n + 1) / 2
3 print("La suma de los primeros números enteros desde 1 hasta " + str(n) + " es " + str(suma))
```

- Escribir un programa que pida al usuario dos números enteros y muestre por pantalla la $<n>$ entre $<m>$ da un cociente $<c>$ y un resto $<r>$ donde $<n>$ y $<m>$ son los números introducidos por el usuario, y $<c>$ y $<r>$ son el cociente y el resto de la división entera respectivamente.

Solución #1



```
1 a=int(input("ingresa un numero para dividir "))
2 b=int(input("ingresa un numero para hacer la division "))
3 print(a,"entre",b,"es igual a",(a/b),"con un resto de",(a%b))
```

Solución #2



```
1 n = input("Introduce el dividendo (entero): ")
2 m = input("Introduce el divisor (entero): ")
3 print(n + " entre " + m + " da un cociente " + str(int(n) // int(m)) + " y un resto " + str(int(n) % int(m)))
```

- Escribir un programa que pregunte al usuario una cantidad a invertir, el interés anual y el número de años, y muestre por pantalla el capital obtenido en la inversión.

Solución #1

```
● ● ●
1 invertir=float(input("ingresa un numero a invertir "))
2 interes=float(input("ingresa el porcentaje de interes anual "))
3 anos=int(input("ingresa la cantidad de anos"))
4 print ("capital final ",round(invertir*(interes/100+1)**anos),2)
```

Solución #2

```
● ● ●
1 cantidad = float(input("¿Cantidad a invertir? "))
2 interes = float(input("¿Interés porcentual anual? "))
3 años = int(input("¿Años?"))
4 print("Capital final: " + str(round(cantidad * (interes / 100 + 1) ** años, 2)))
```

- Una juguetería tiene mucho éxito en dos de sus productos: payasos y muñecas. Suele hacer venta por correo y la empresa de logística les cobra por peso de cada paquete así que deben calcular el peso de los payasos y muñecas que saldrán en cada paquete a demanda. Cada payaso pesa 112 g y cada muñeca 75 g. Escribir un programa que lea el número de payasos y muñecas vendidos en el último pedido y calcule el peso total del paquete que será enviado.

Solución #1

```
● ● ●
1 pp=112
2 pm=75
3 p=int(input("ingresa el numero de payasos que deseas: "))
4 m=int(input("ingresa el numero de muñecas que deseas: "))
5 pt=p*pp+m*pm
6 print("el total de paquetes en gramos es: ",pt)
```

- Imagina que acabas de abrir una nueva cuenta de ahorros que te ofrece el 4% de interés al año. Estos ahorros debido a intereses, que no se cobran hasta finales de año, se te añaden al balance final de tu cuenta de ahorros. Escribir un programa que comience leyendo la cantidad de dinero depositada en la cuenta de ahorros, introducida por el usuario. Después el programa debe calcular y mostrar por pantalla la cantidad de ahorros tras el primer, segundo y tercer años. Redondear cada cantidad a dos decimales.

Solución #1

```

● ● ●

1 deposito=float(input("ingresa el valor del deposito "))
2 interes=0.04
3 ahorro1=deposito*(1+interes)
4 print("el valor del ahorro con intereses del primer año es: ",round(ahorro1,2))
5 ahorro2=ahorro1*(1+interes)
6 print("el valor del ahorro con intereses del segundo año es: ",round(ahorro2,2))
7 ahorro3=ahorro2*(1+interes)
8 print("el valor del ahorro con intereses del tercer año es: ",round(ahorro3,2))

```

- Una panadería vende barras de pan a 3.49€ cada una. El pan que no es el día tiene un descuento del 60%. Escribir un programa que comience leyendo el número de barras vendidas que no son del día. Después el programa debe mostrar el precio habitual de una barra de pan, el descuento que se le hace por no ser fresca y el coste final total.

Solución #1

```

● ● ●

1 ventas_atrasadas=int(input("ingresa las ventas atrasadas: "))
2 precio_habitual=3.49
3 descuento=0.60*(ventas_atrasadas*precio_habitual)
4 print("precio habitual del pan es de: ",round(precio_habitual*ventas_atrasadas,2))
5 print("total de descuento: ",round(descuento,2))
6 print("total a pagar: ",round(precio_habitual*ventas_atrasadas-descuento,2))

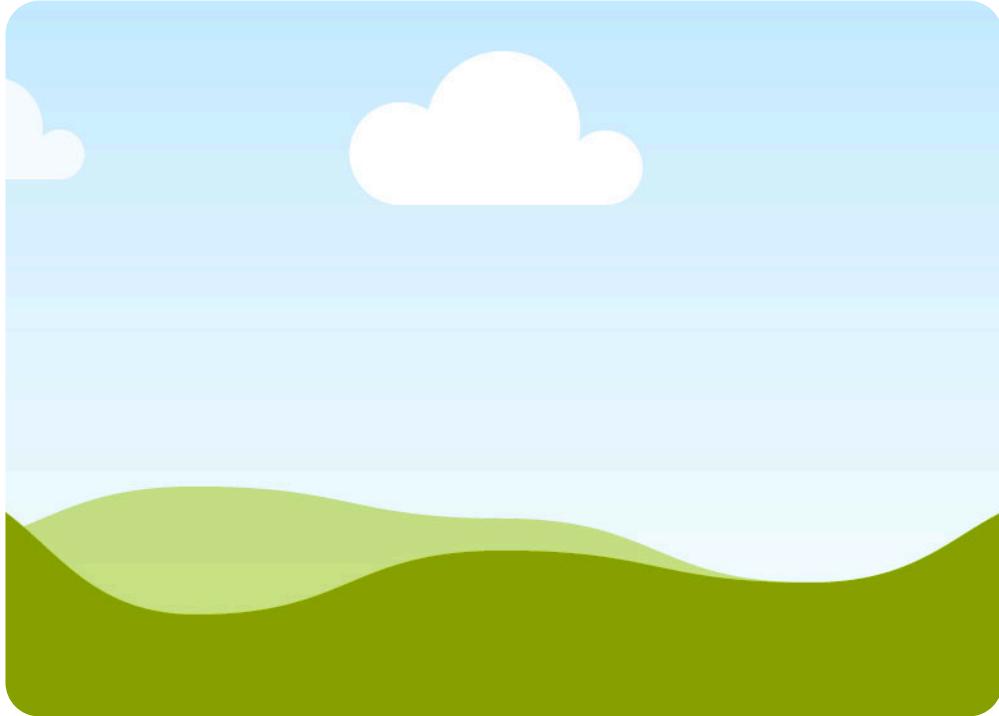
```

CAPITULO 4: Control de cadenas

CADENAS

SE BASA EN:

Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE OPERADORES

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

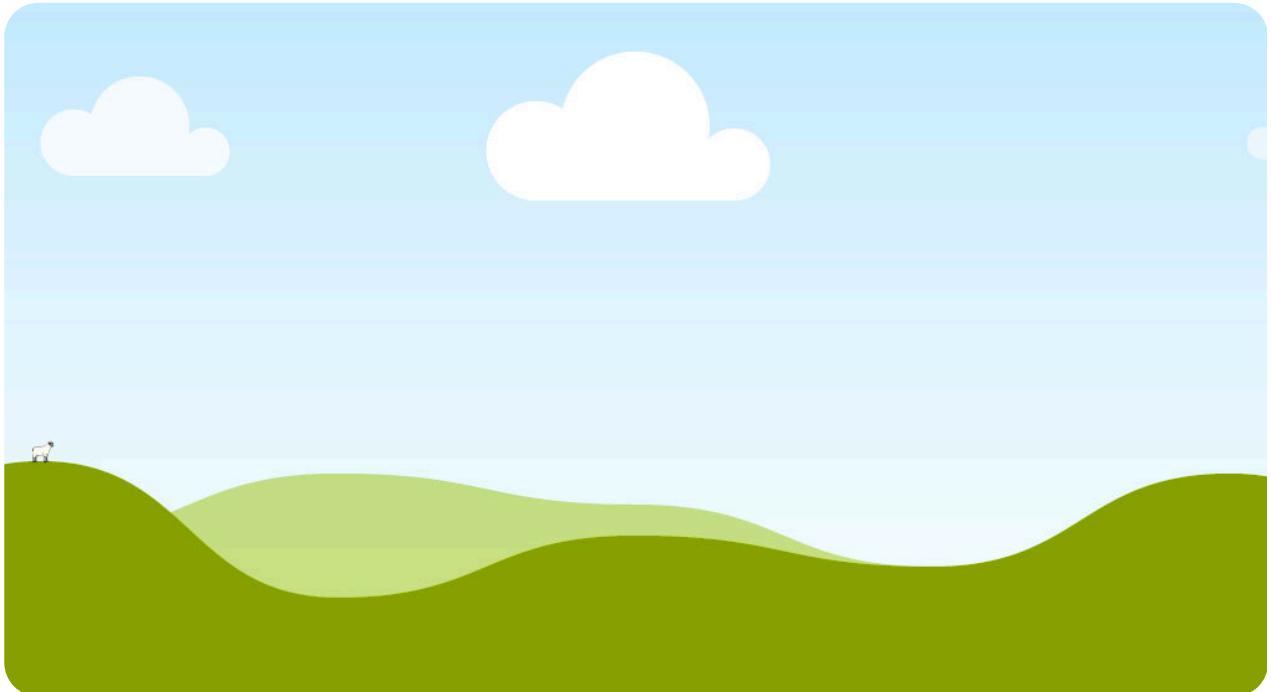
```
1 a=10 #variable de tipo entero
2 b=2 #variable de tipo entero
3 suma=a+b
4 resta=a-b
5 multiplicacion=a*b
6 division=a/b
7 cociente=a//b
8 residuo=a%b
9 potencia=a**b
10
11 print('este es el valor de la suma',suma) #resultado es 12
12 print('este es el valor de la resta',resta) #resultado es 8
13 print('este es el valor de la multiplicacion',multiplicacion )#resultado es 20
14 print('este es el valor de la division ',division) #resultado es 5
15 print('este es el valor de la cociente',cociente) #resultado es 5
16 print('este es el valor de la residuo',residuo) #resultado es 0
17 print('este es el valor de la potencia',potencia) #resultado es 100
```

CAPITULO 5: Flujo de Condicionales

CONDICIONALES

SE BASA EN:

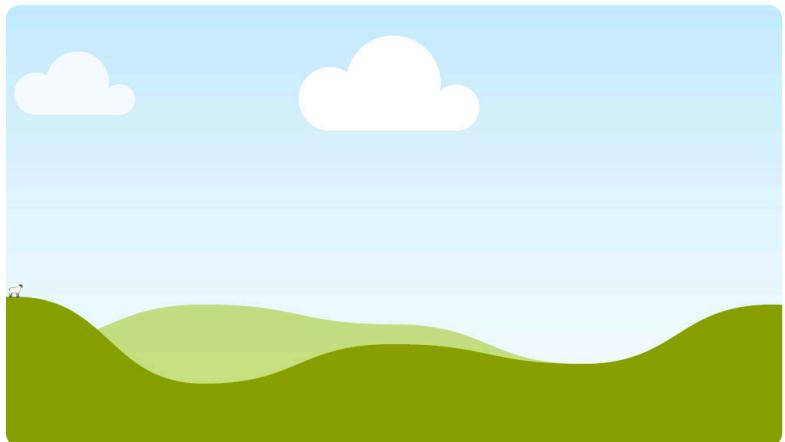
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

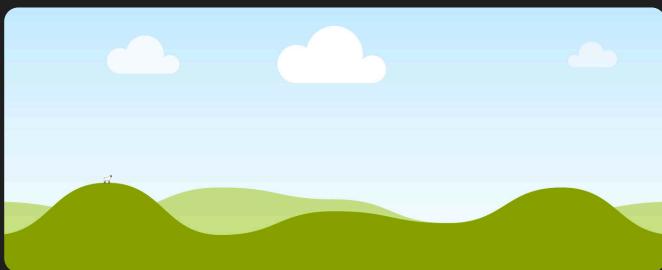


EJERCICIOS DE PRACTICA

BLOQUE III

- Construir un programa que reciba dos números enteros y calcule el resultado de elevar el 1º número leído a la potencia representada por el 2º número leído

Solución #1

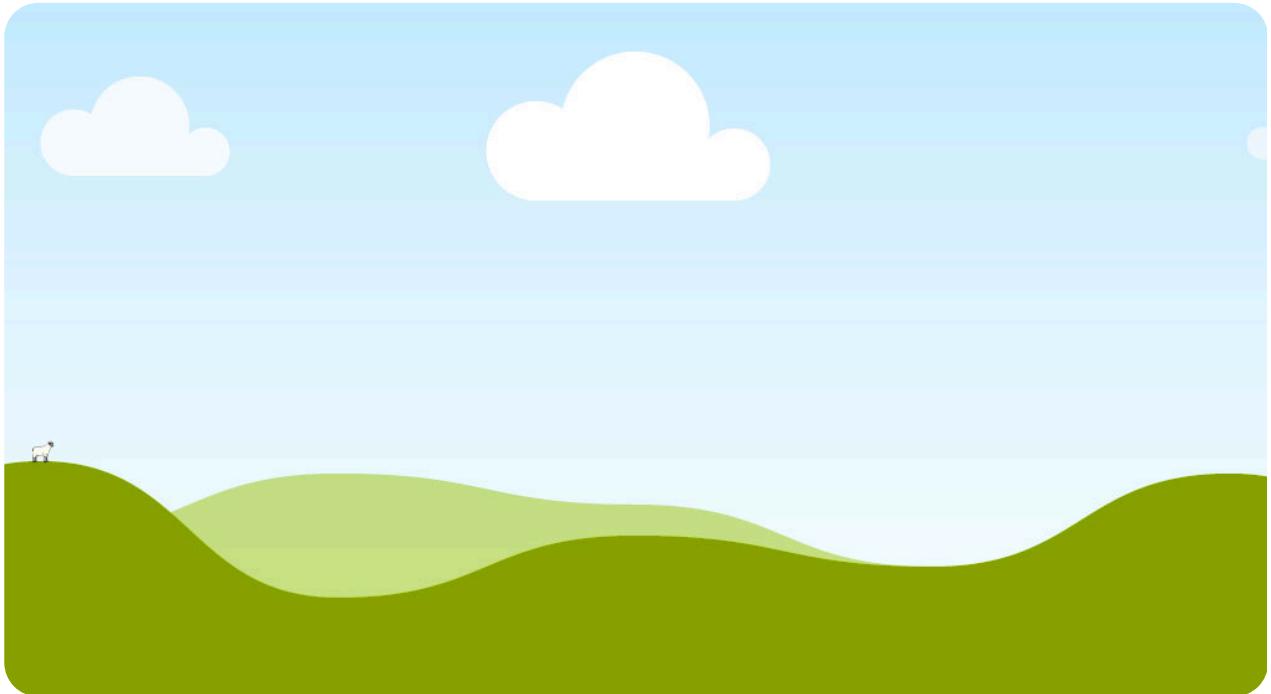


CAPITULO 6: Bucles

BUCLLES

SE BASA EN:

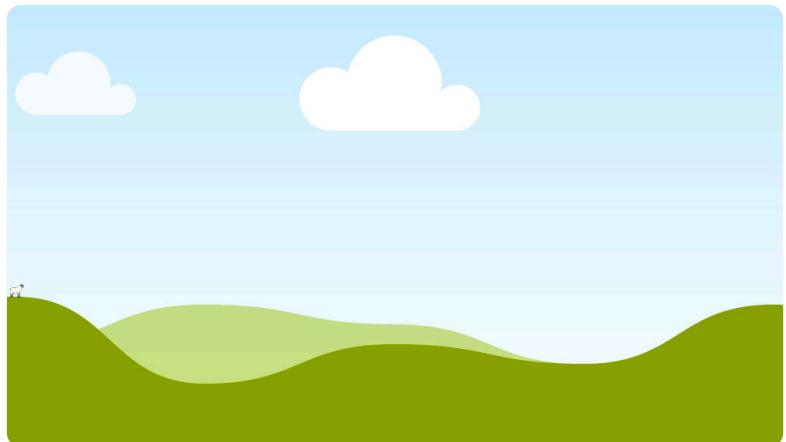
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

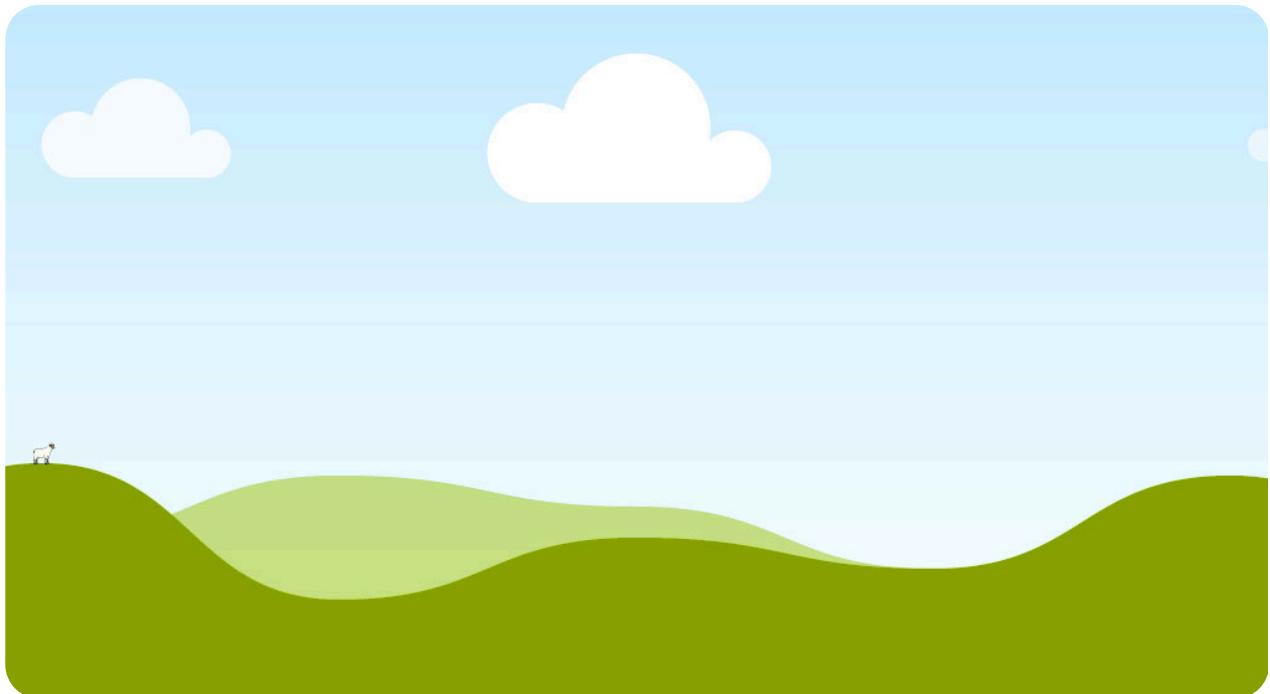


CAPITULO 7: Manejo de Listas y Tuplas

LISTAS Y TUPLAS

SE BASA EN:

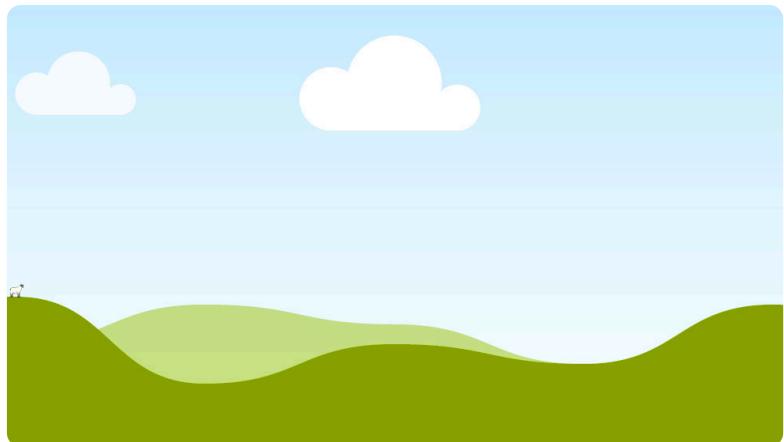
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

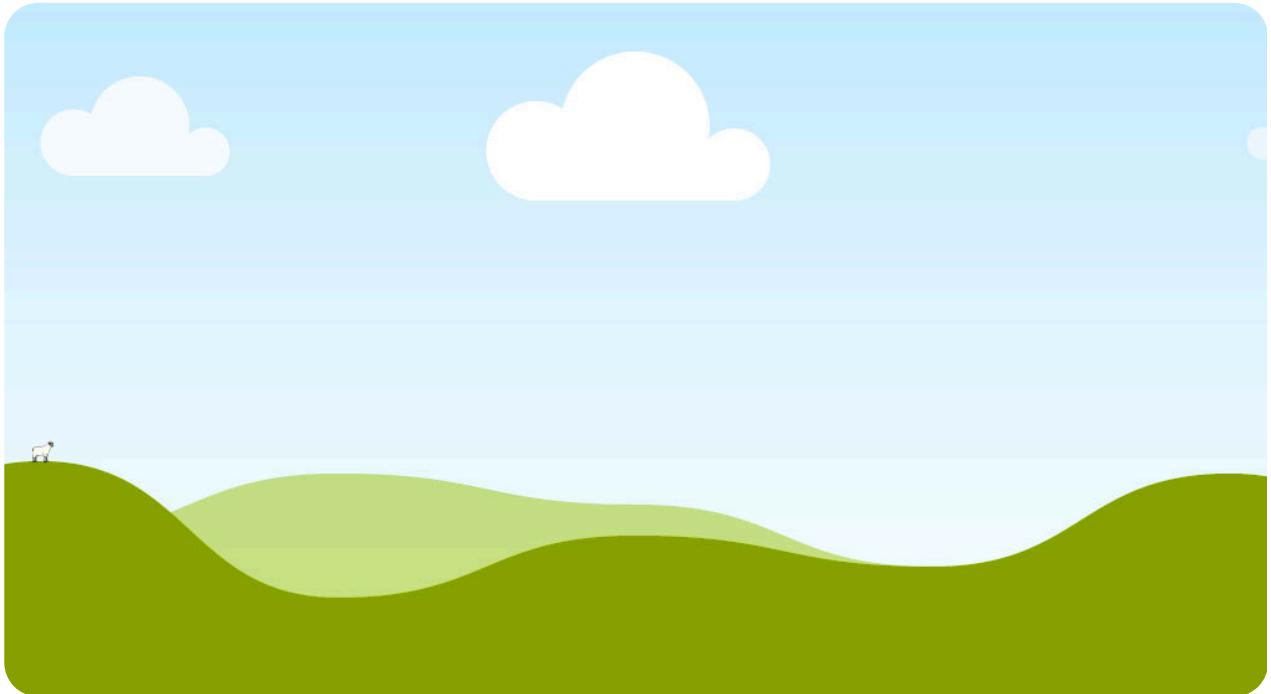


CAPITULO 8: Funciones

FUNCIONES

SE BASA EN:

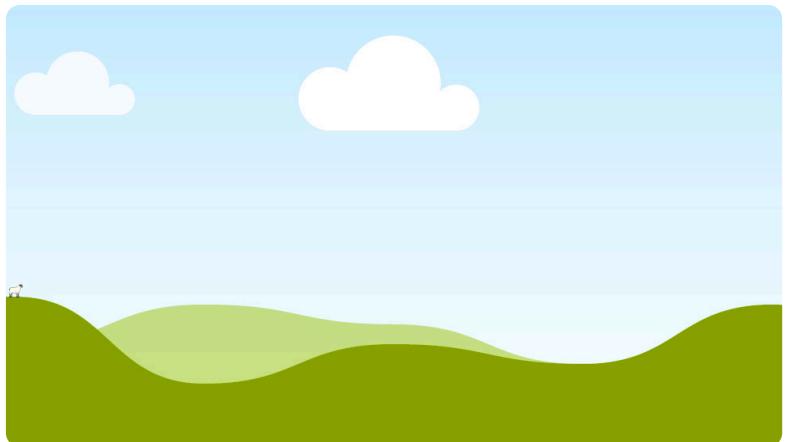
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

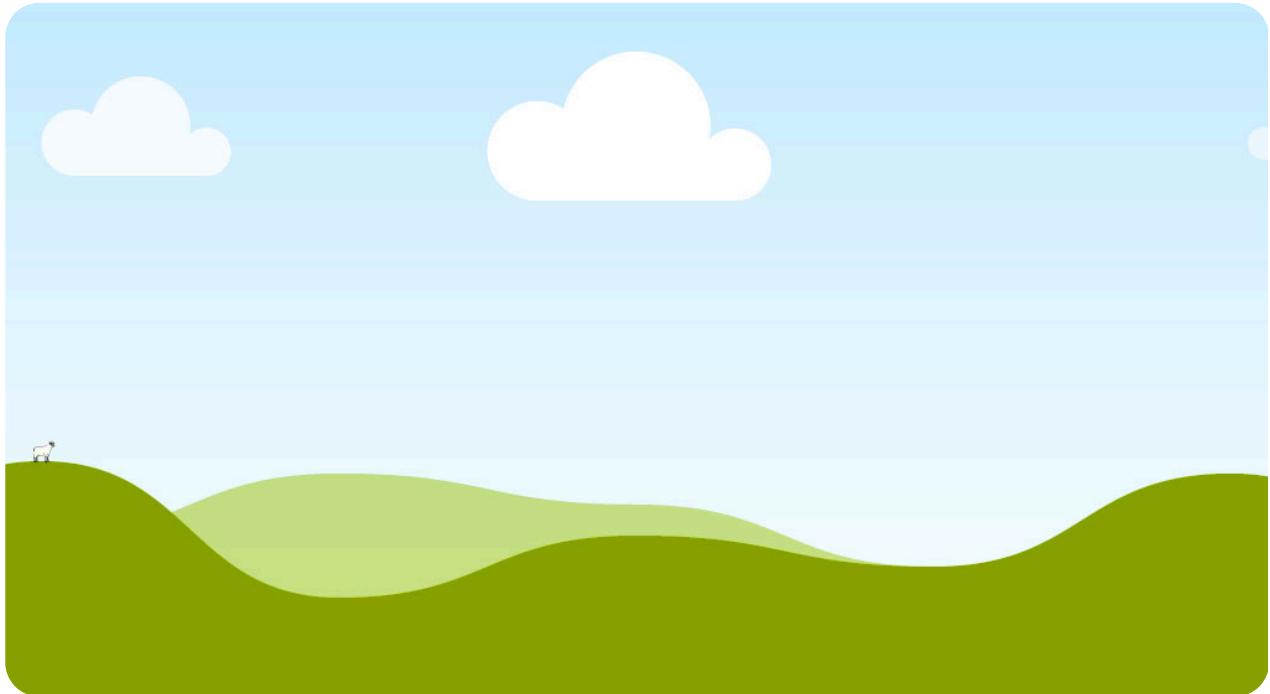


CAPITULO 9: Generadores

GENERADORES

SE BASA EN:

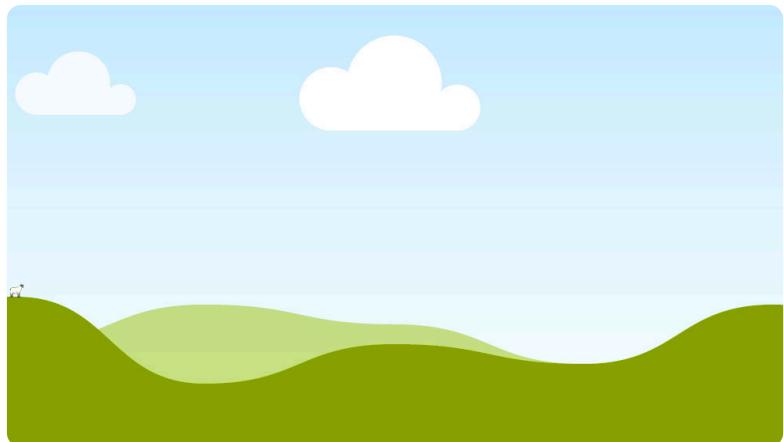
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

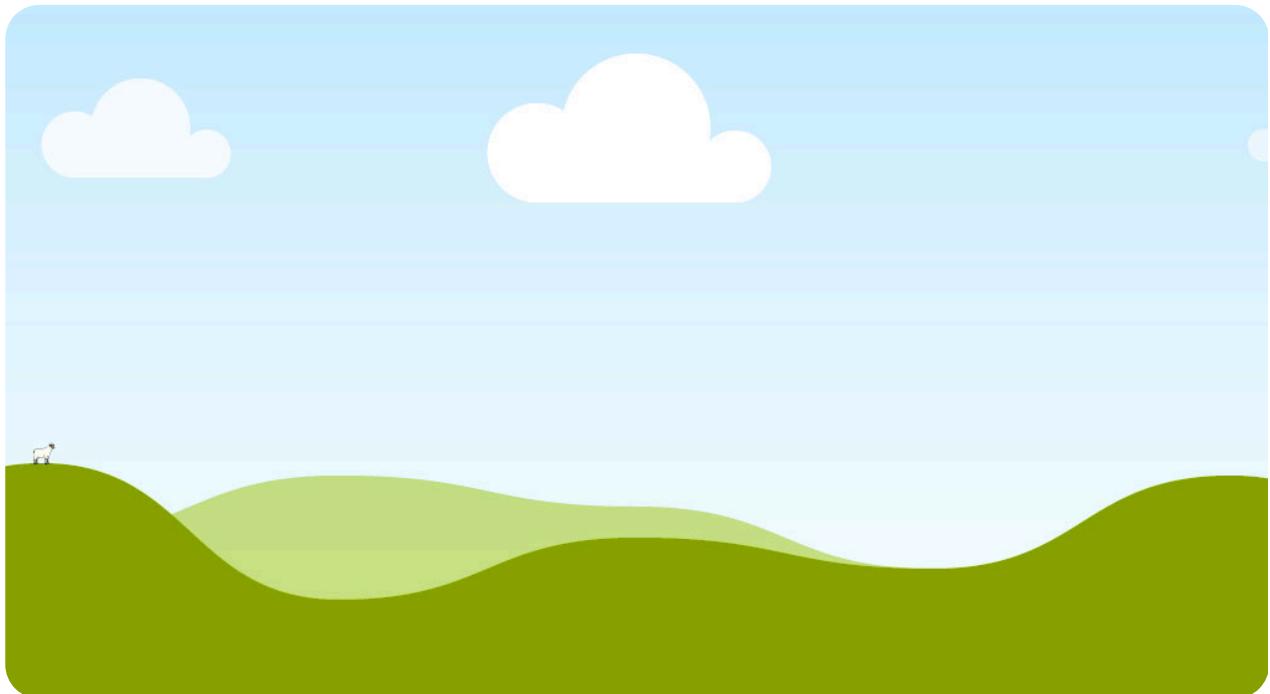


CAPITULO 10: Diccionarios

DICCIONARIOS

SE BASA EN:

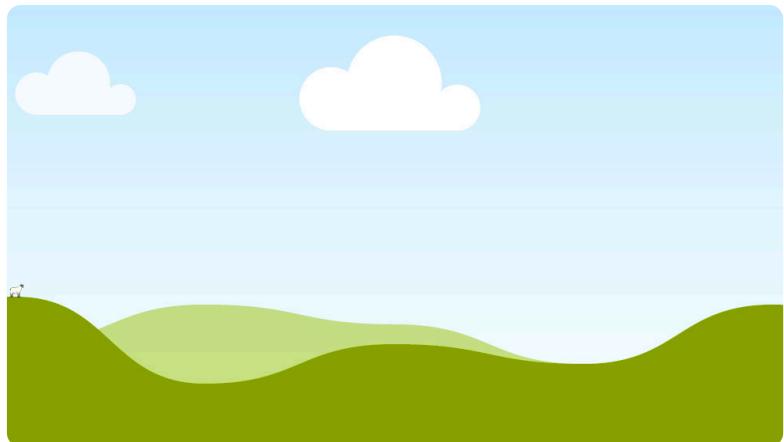
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

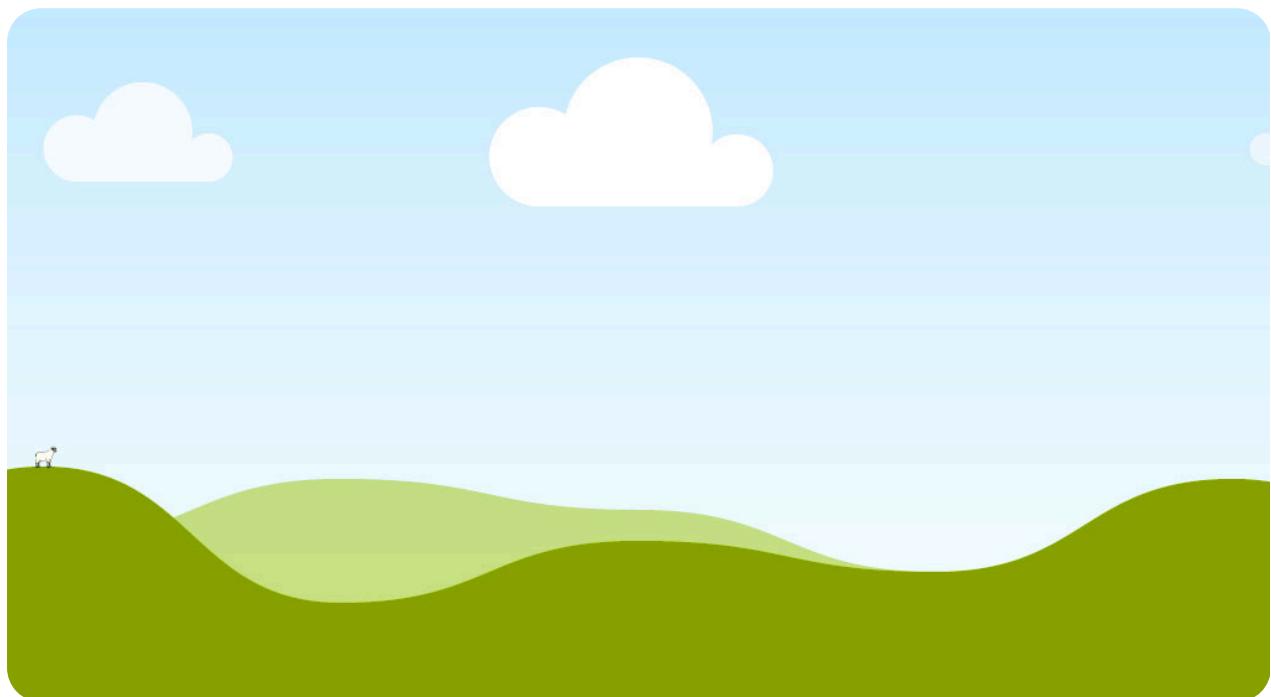


CAPITULO 11: Programación Funcional

PROGRAMACIÓN FUNCIONAL

SE BASA EN:

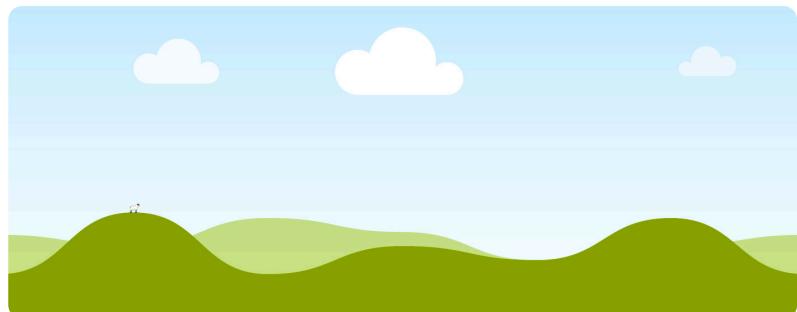
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

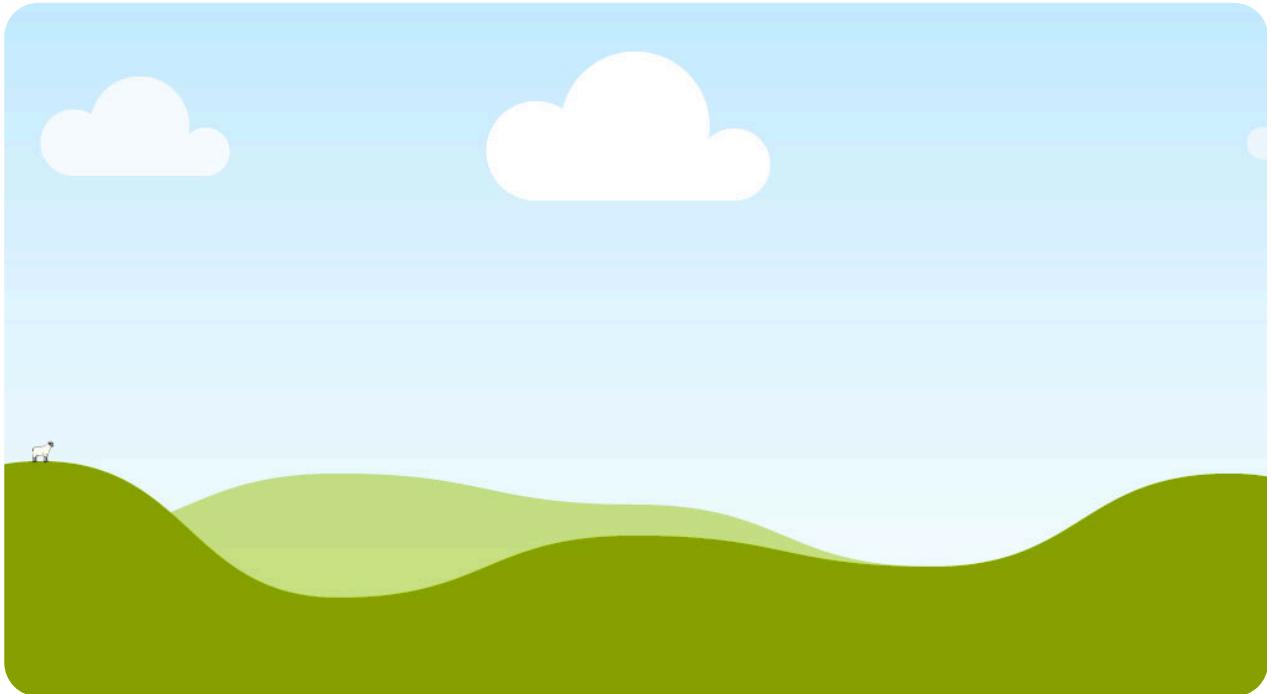


CAPITULO 12: Ficheros

FICHEROS

SE BASA EN:

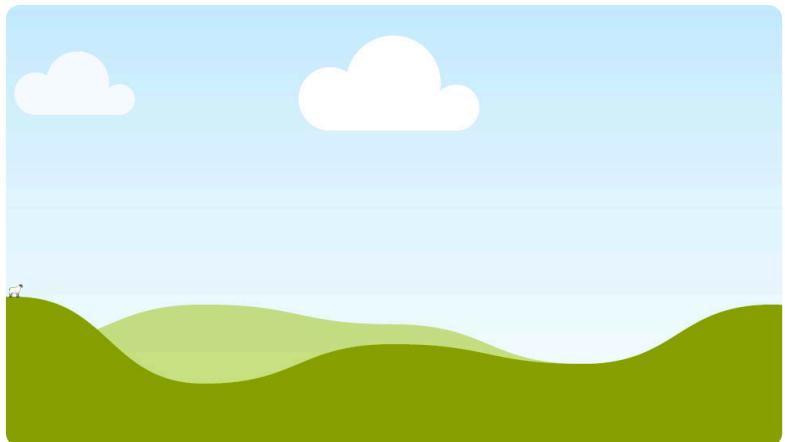
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

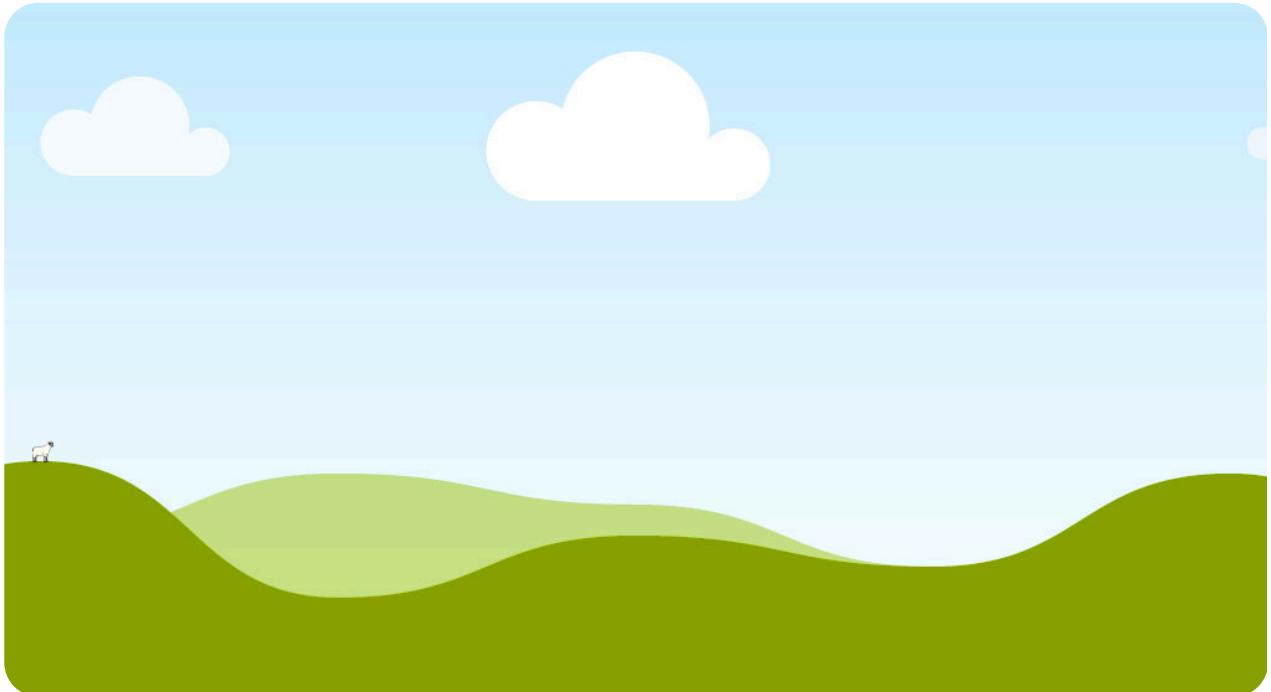


CAPITULO 13: Depuración

DEPURACIÓN

SE BASA EN:

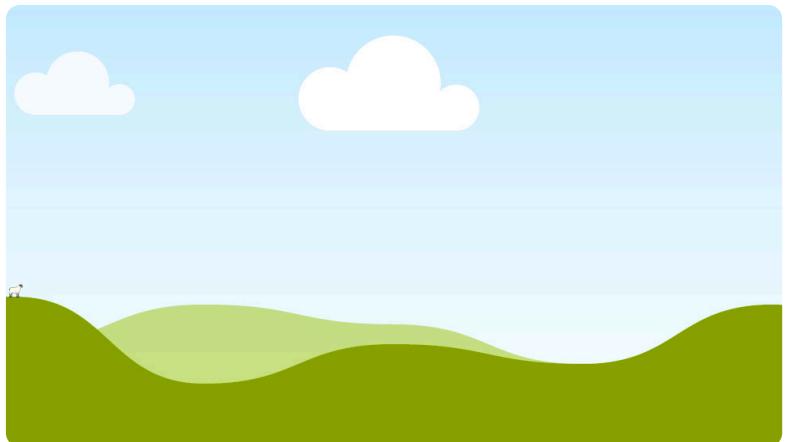
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

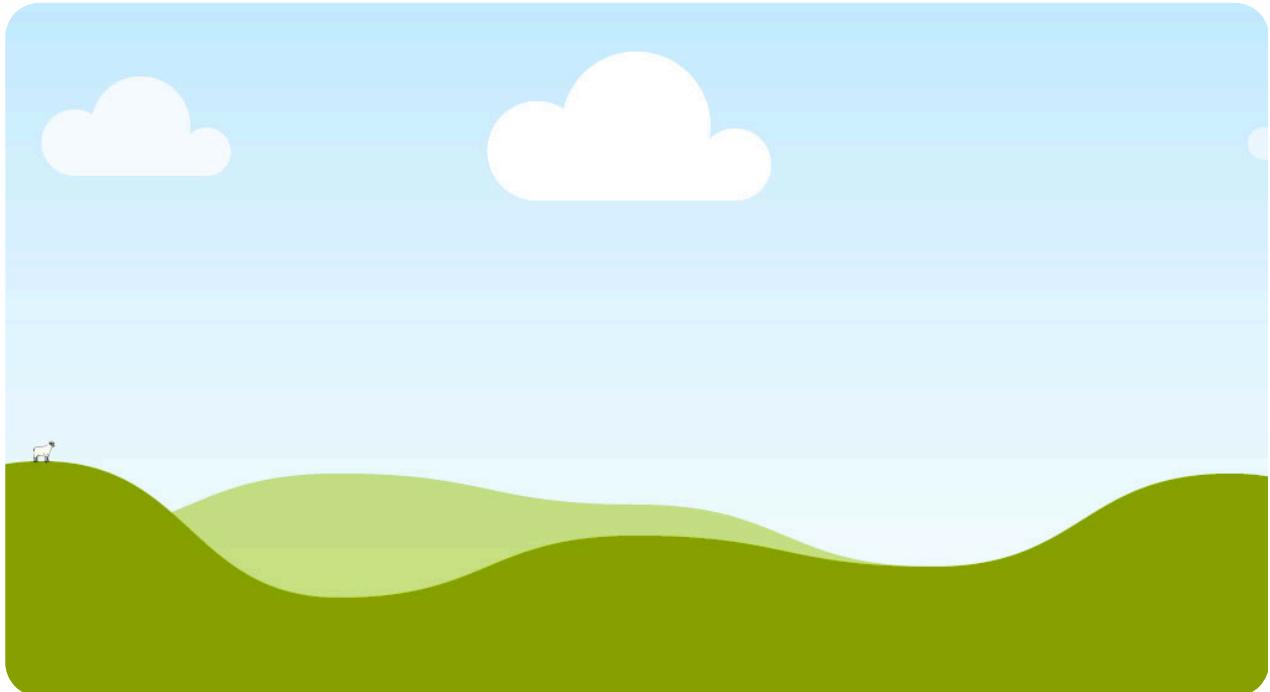


CAPITULO 14: Manejo de Librerías

MANEJO DE LIBRERIA

SE BASA EN:

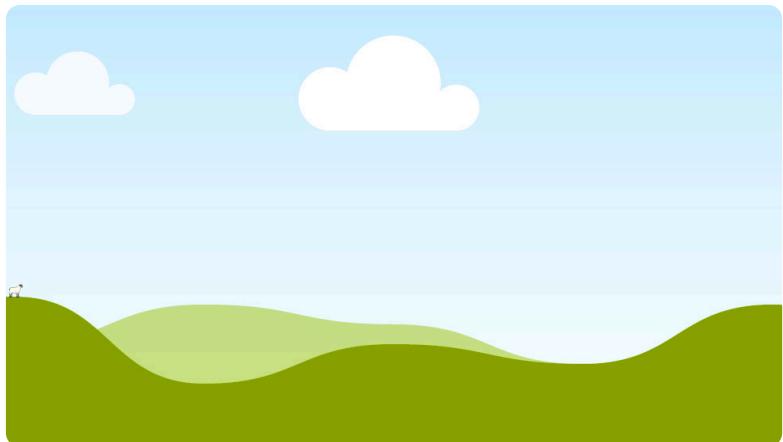
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

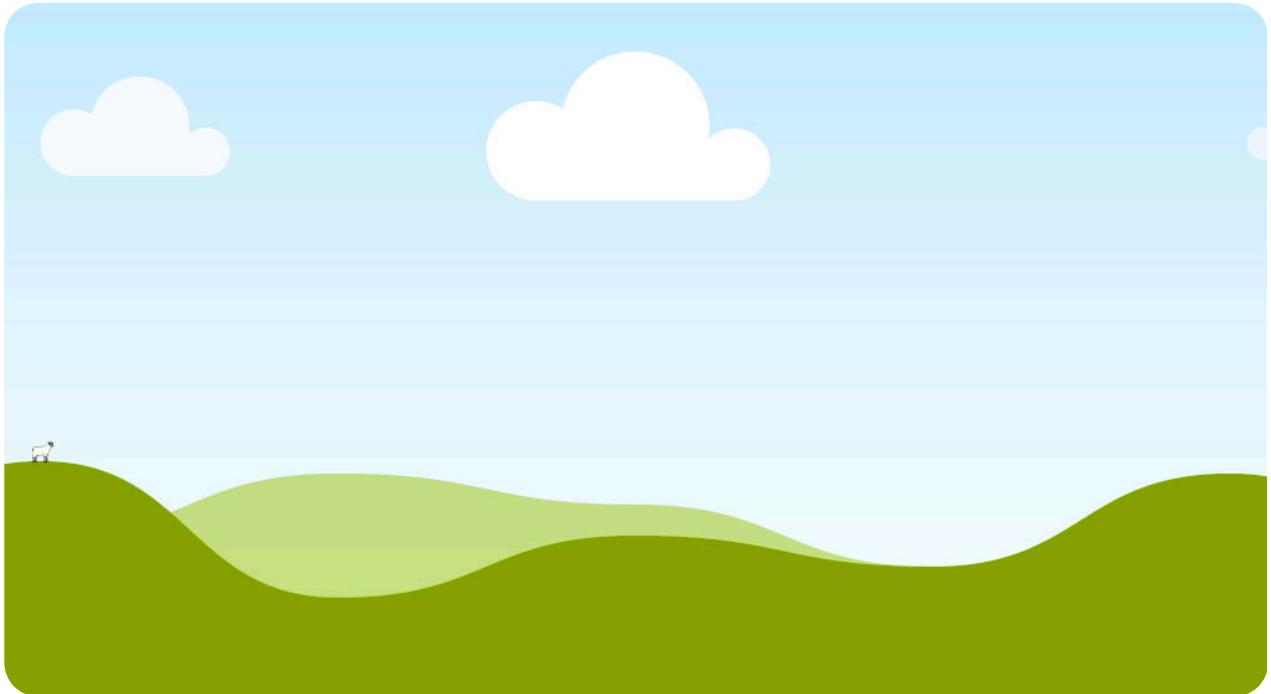


CAPITULO 15: Excepciones

EXCEPCIONES

SE BASA EN:

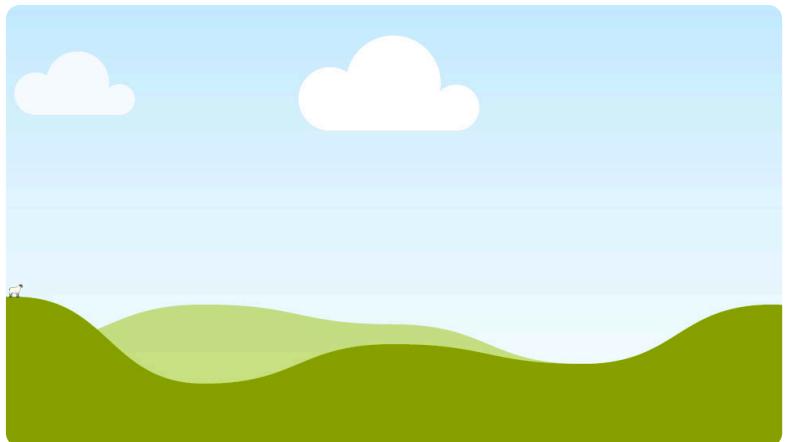
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

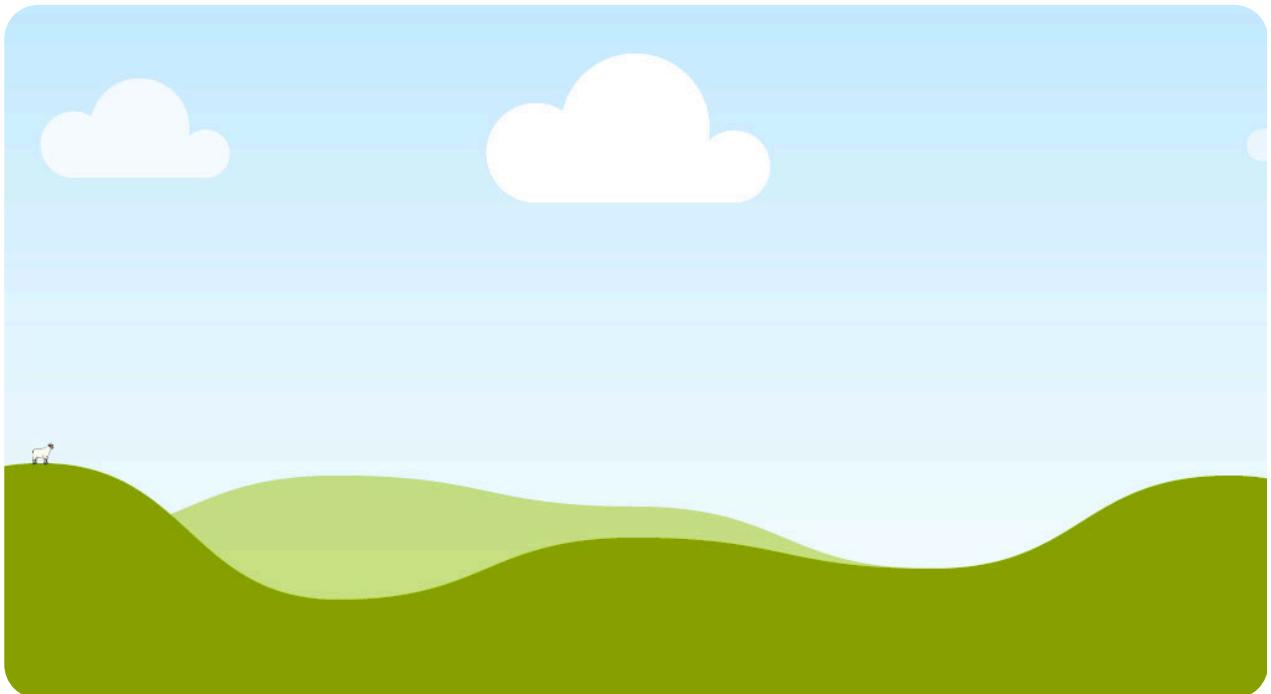


CAPITULO 16: Introducción a POO

P. ORIENTADA A OBJ.

SE BASA EN:

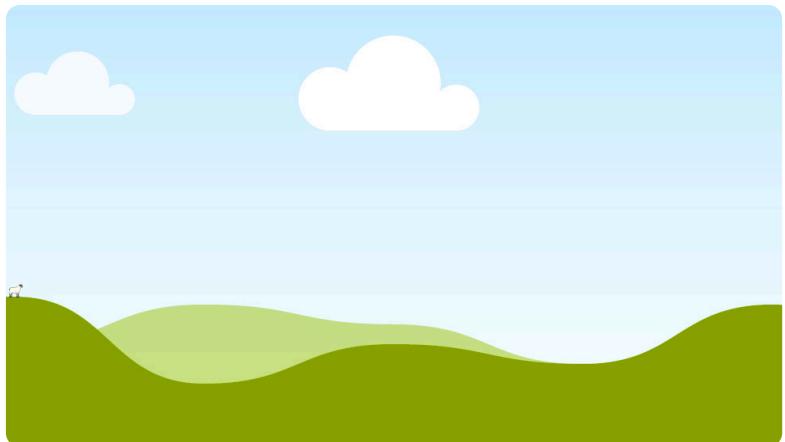
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

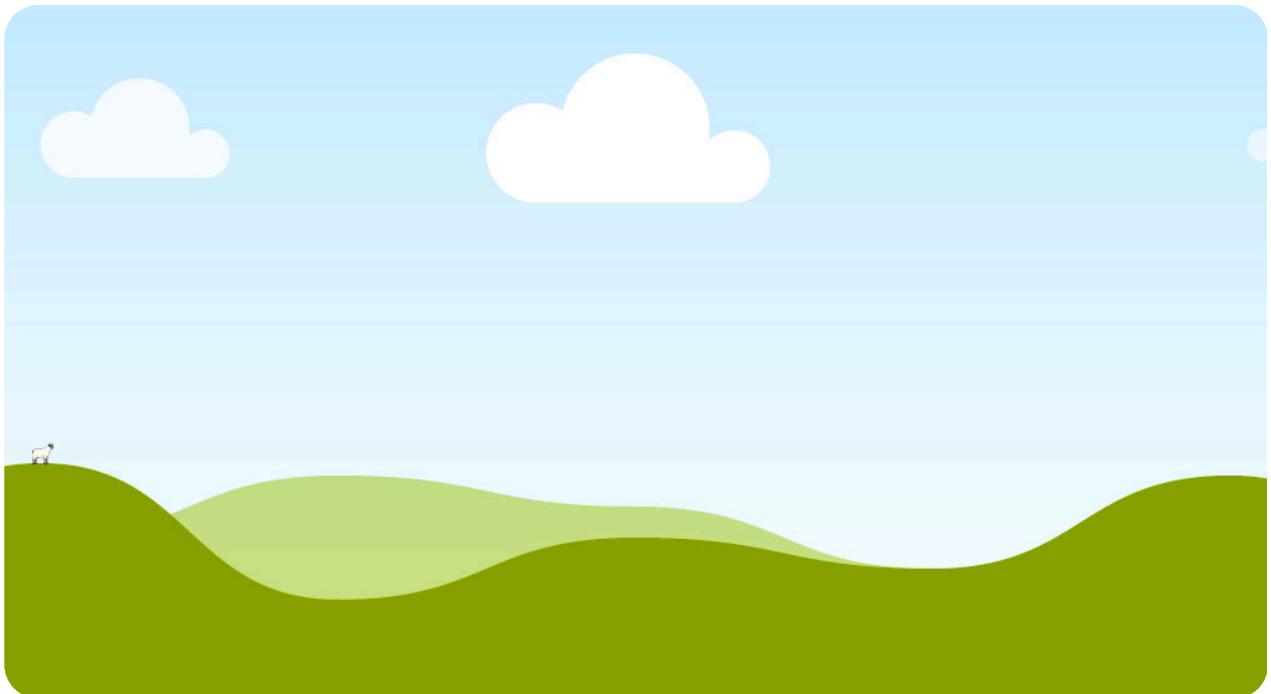


CAPITULO 17: Constructores

CONSTRUCTORES

SE BASA EN:

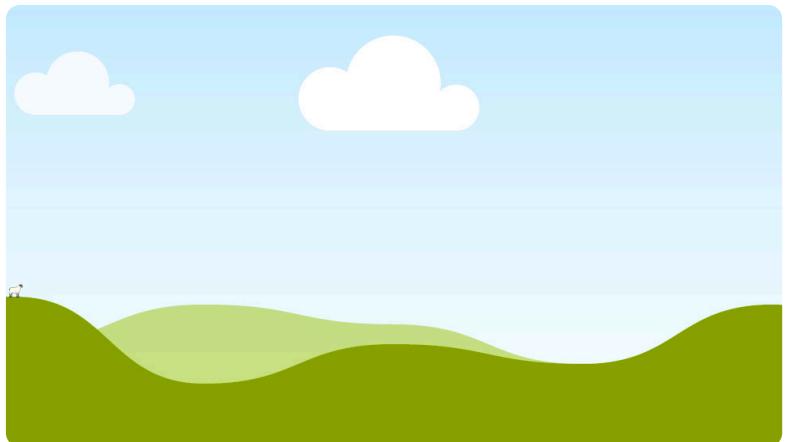
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

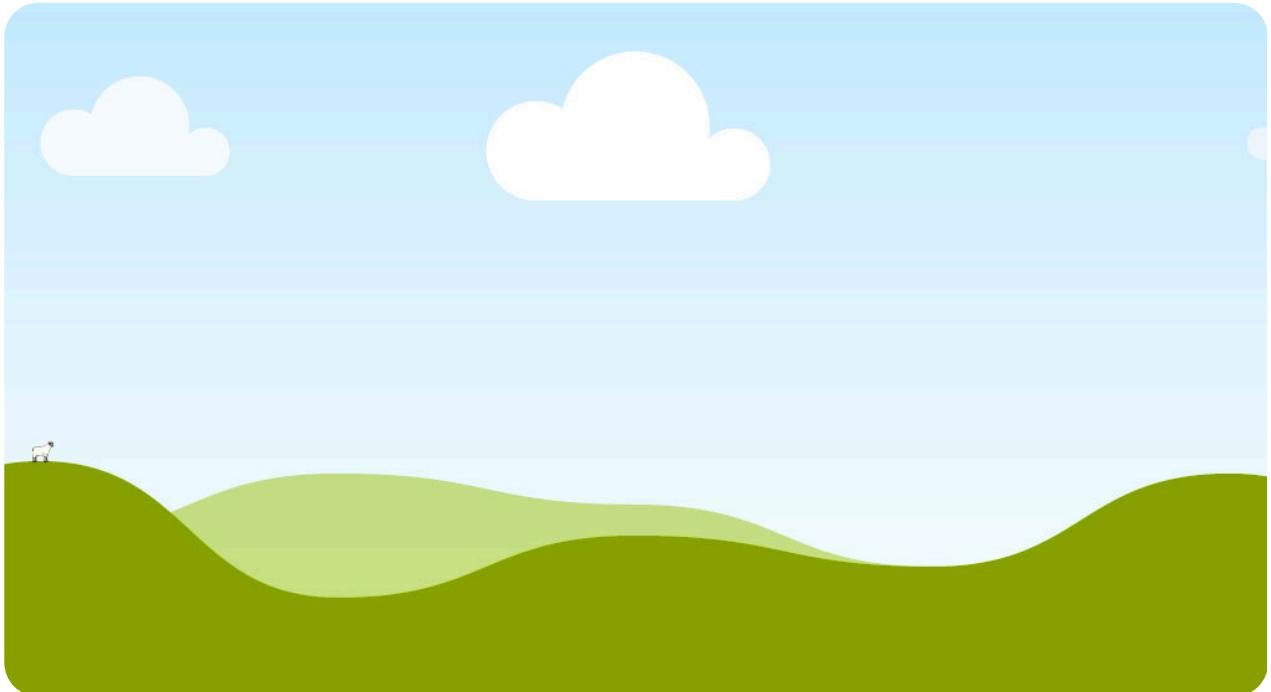


CAPITULO 18: Encapsulamiento

ENCAPSULAMIENTO

SE BASA EN:

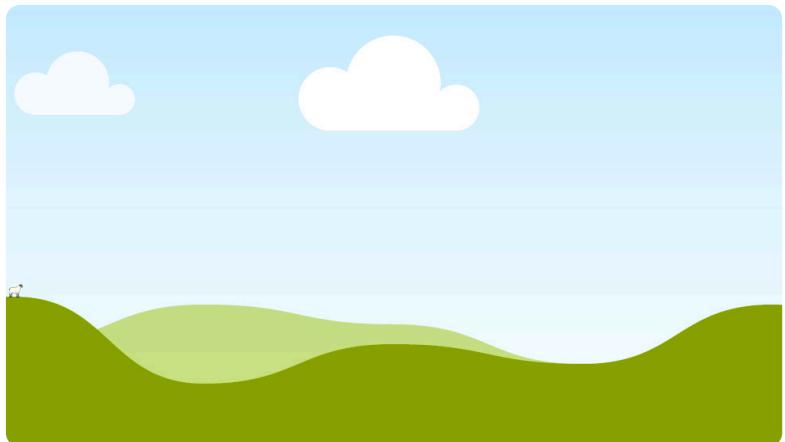
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

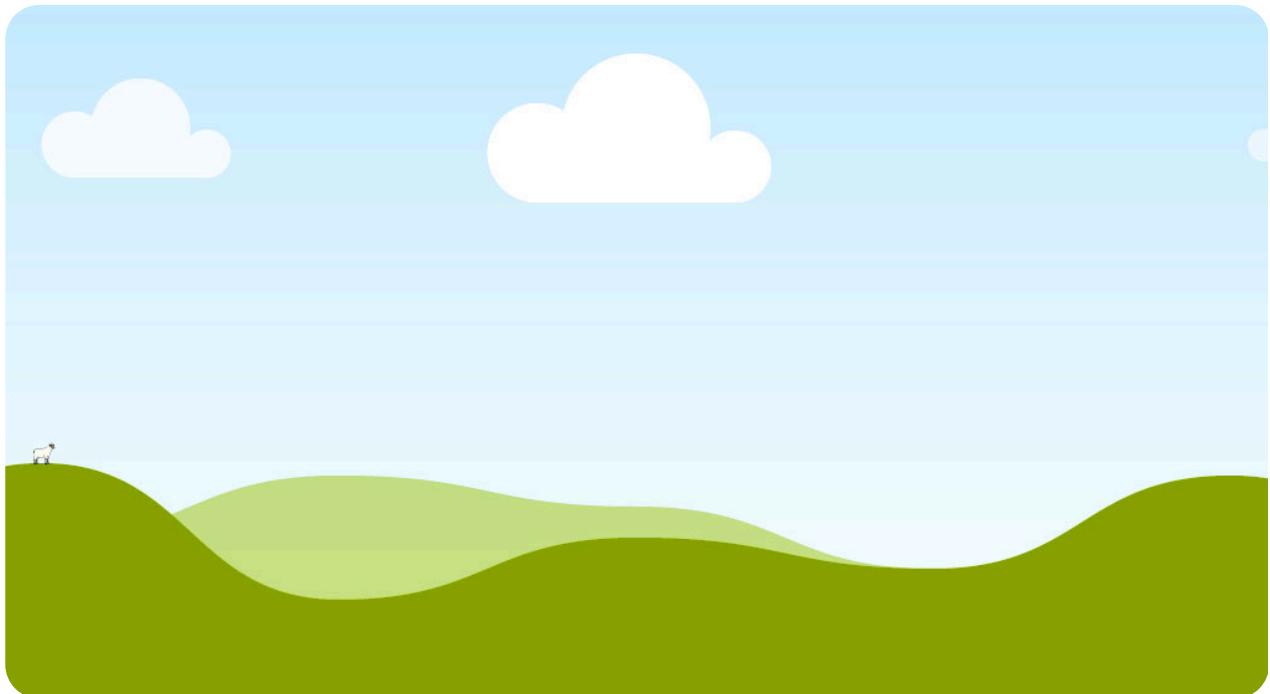


CAPITULO 19: Herencia

HERENCIA

SE BASA EN:

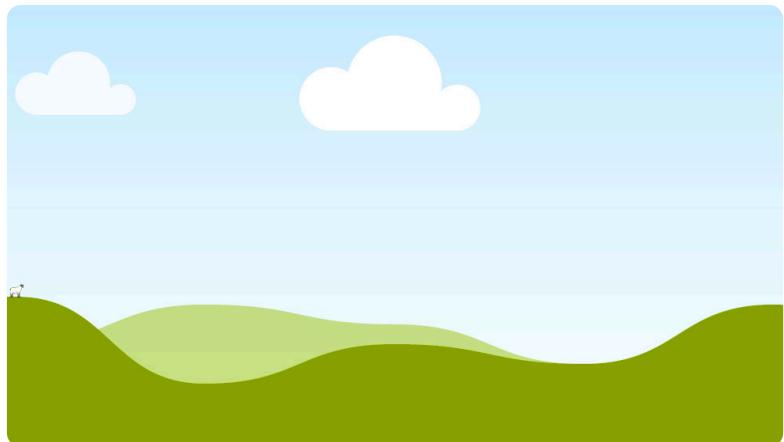
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

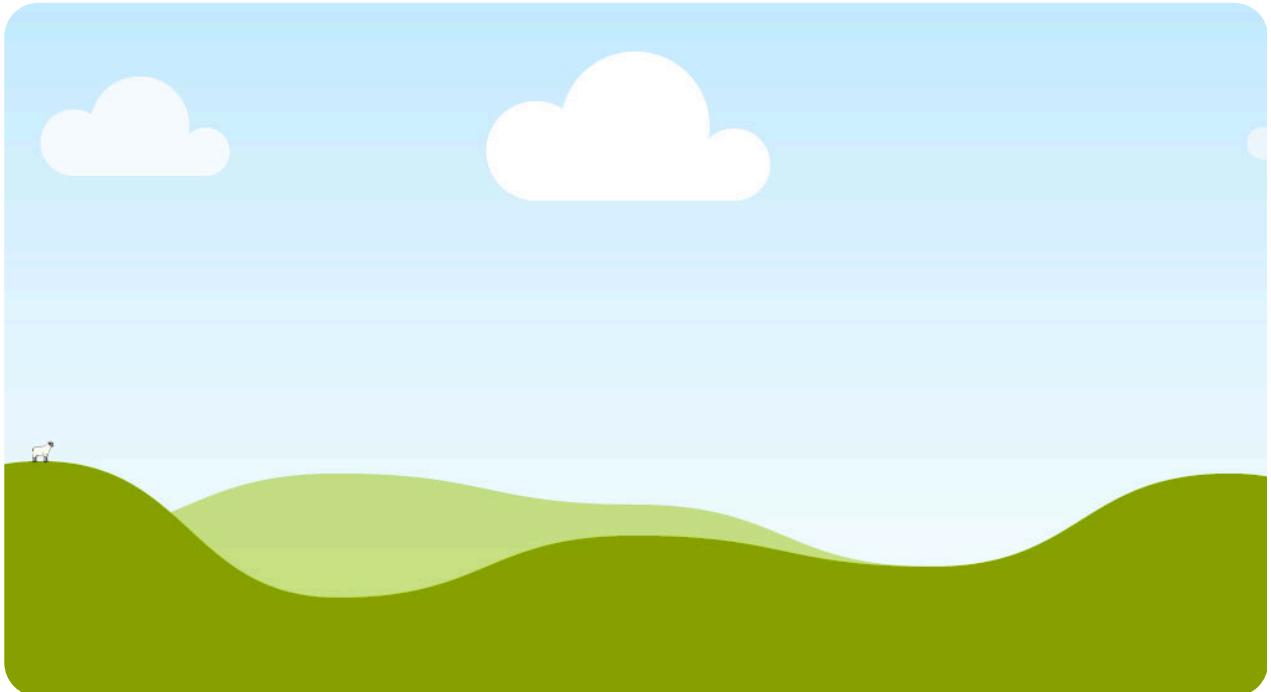


CAPITULO 20: Strings

STRINGS

SE BASA EN:

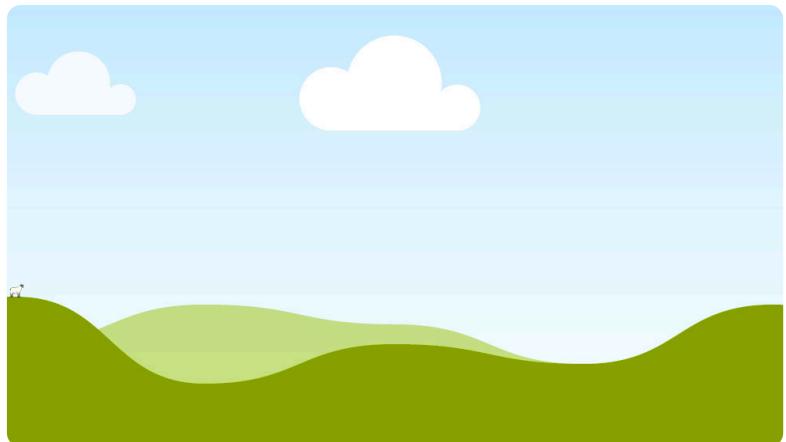
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

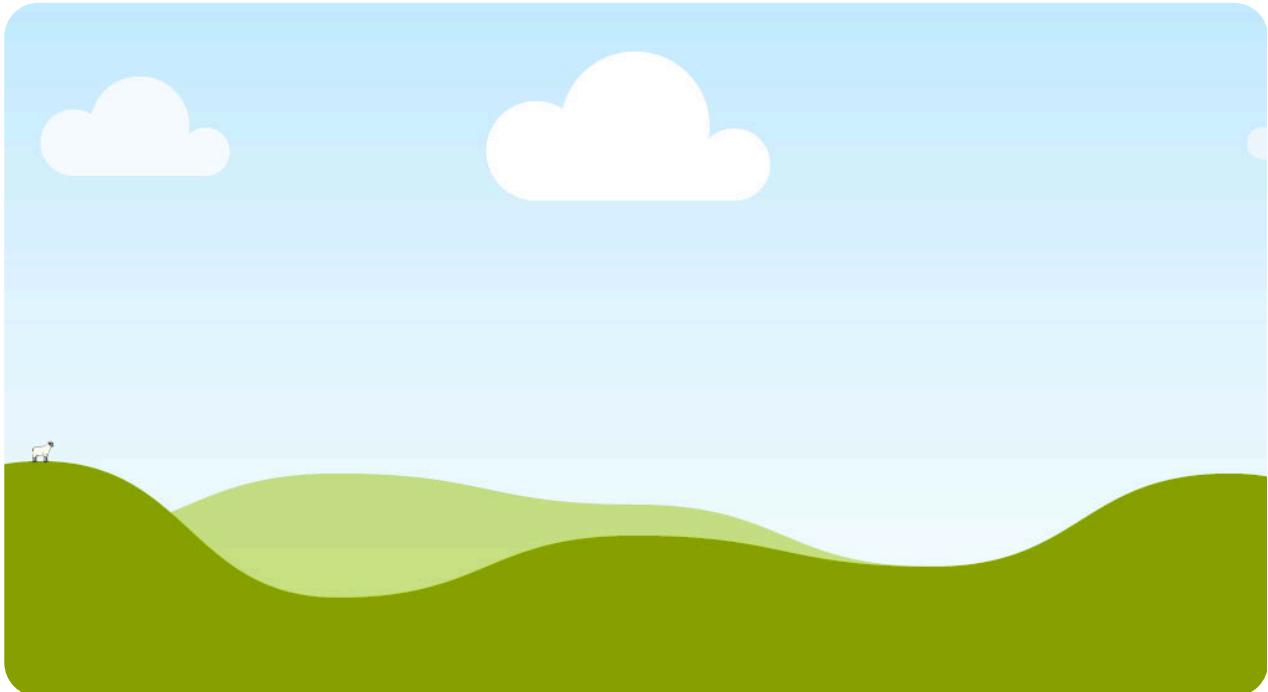


CAPITULO 21: Módulos y Paquetes

MÓDULOS Y PAQUETES

SE BASA EN:

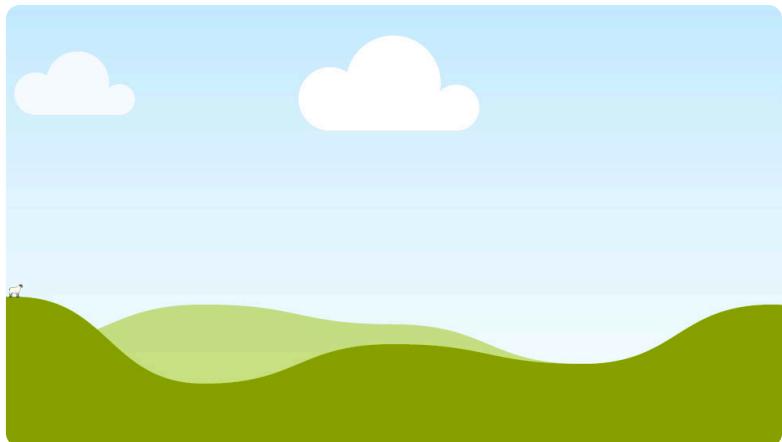
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

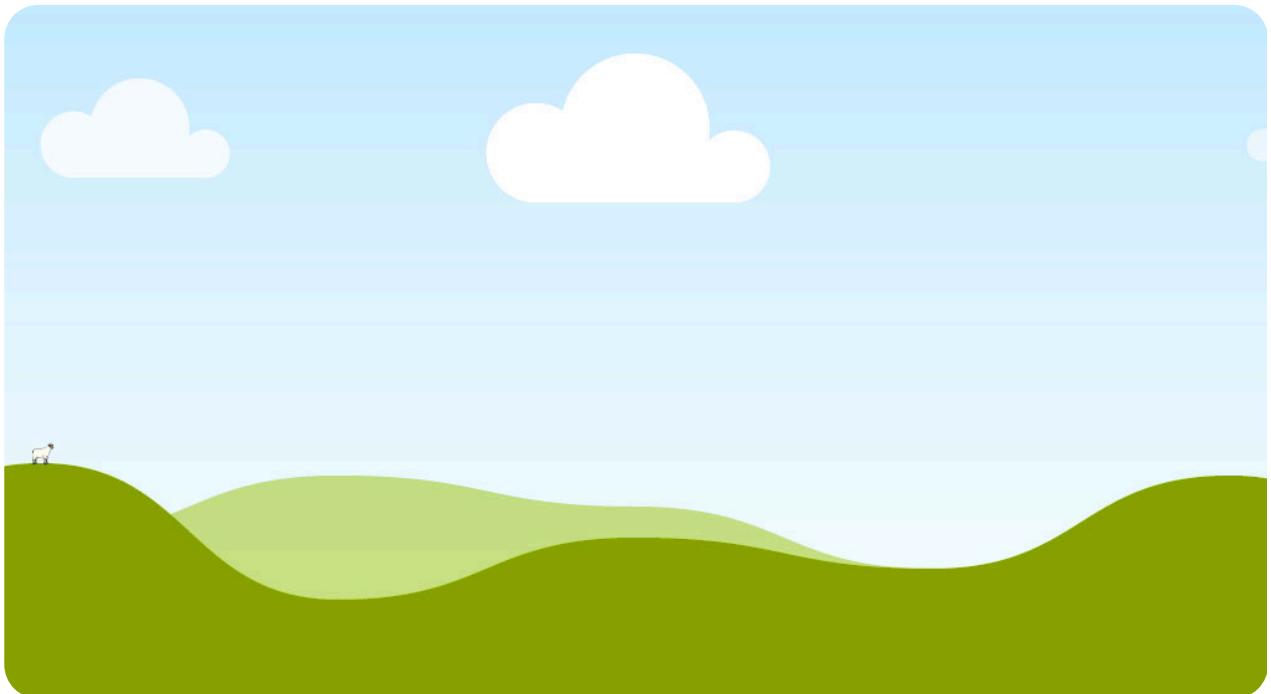


CAPITULO 22: Archivos de Texto

ARCHIVOS DE TEXTO

SE BASA EN:

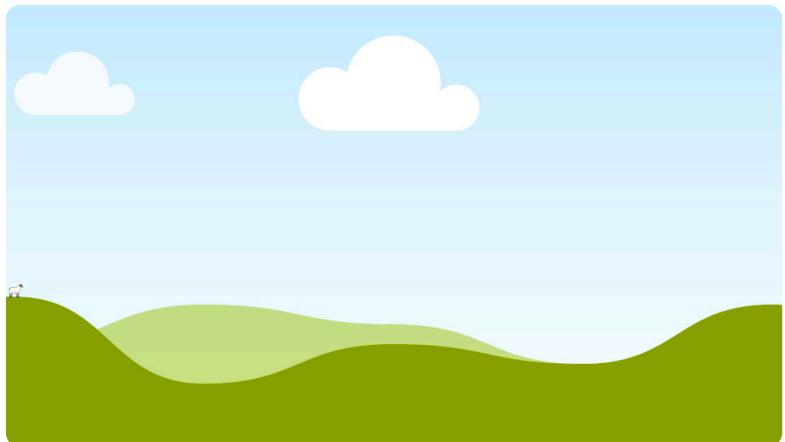
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

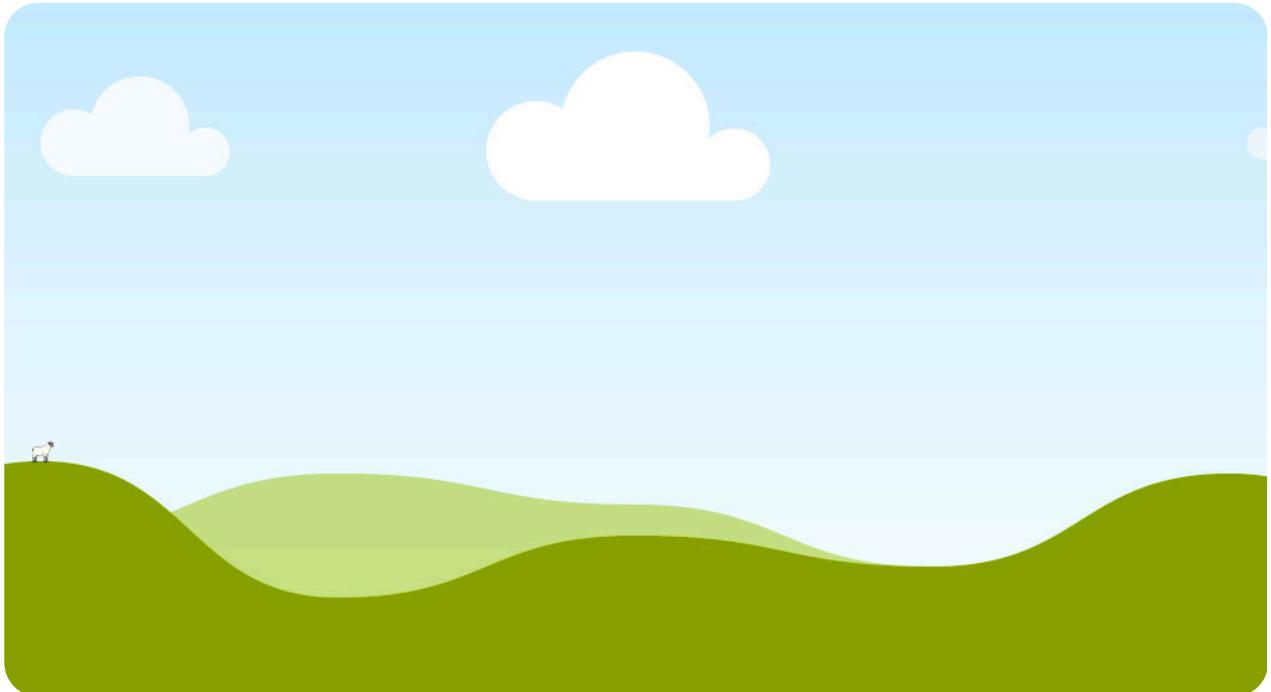


CAPITULO 23: Archivos Binarios

ARCHIVOS BINARIOS

SE BASA EN:

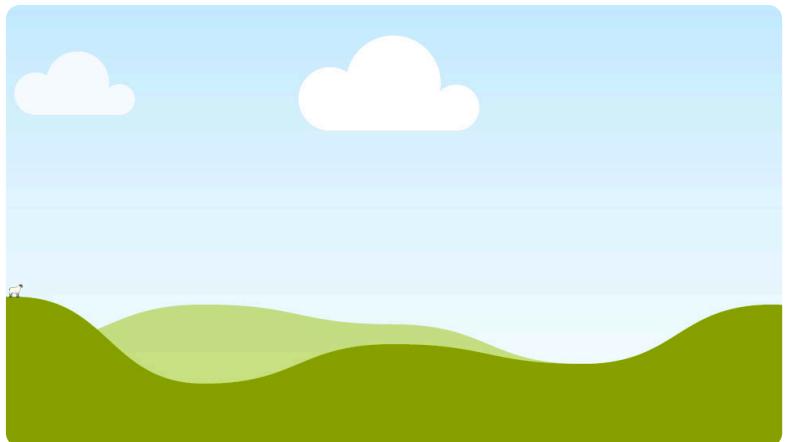
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

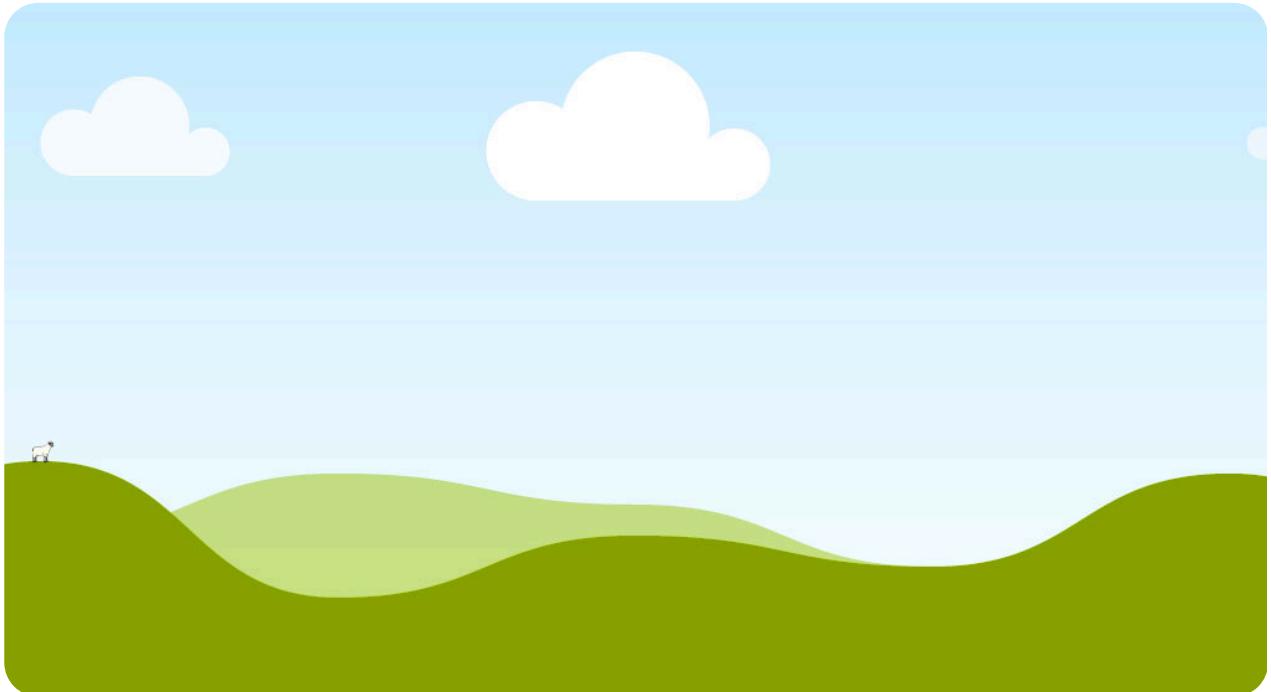


CAPITULO 24: Ventanas

VENTANAS

SE BASA EN:

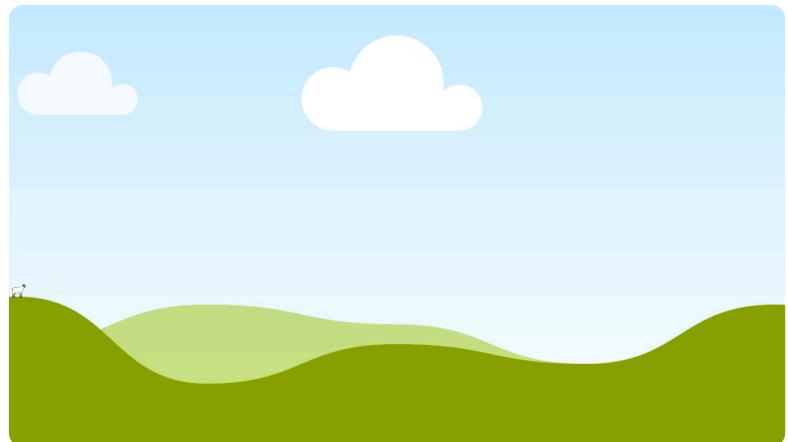
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

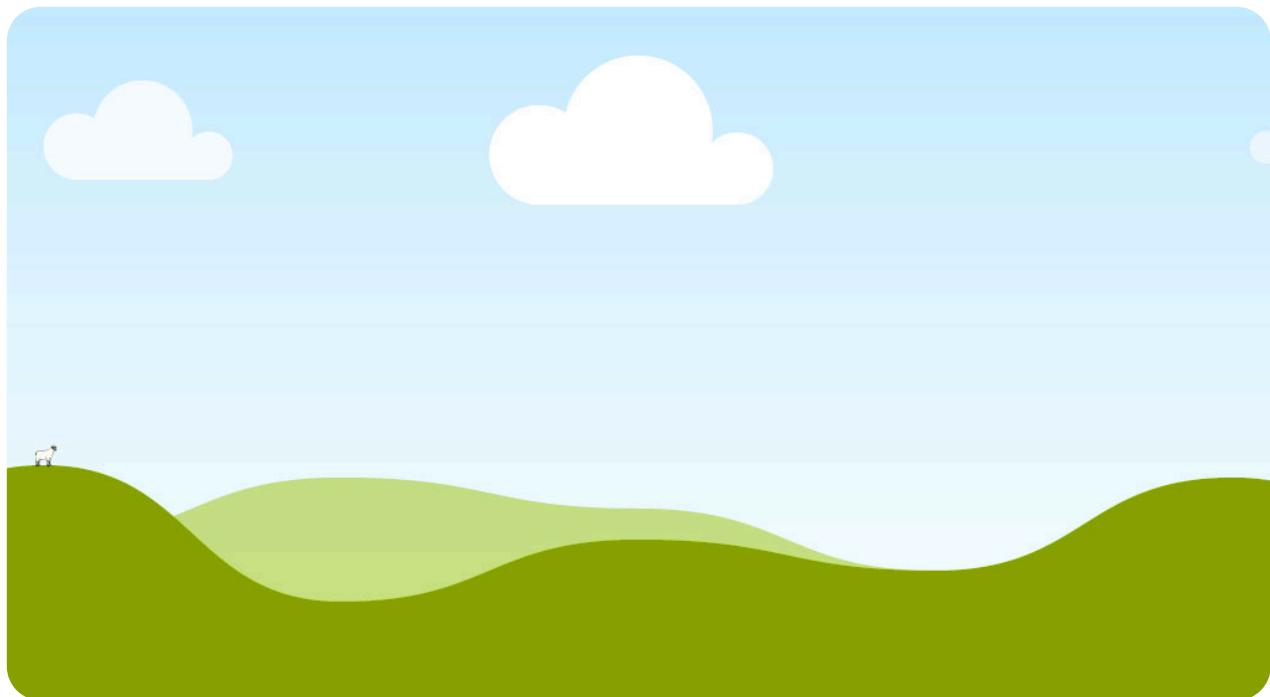


CAPITULO 25: Controlador / Variables GUI

CONTROLADOR Y VARIABLES GUI

SE BASA EN:

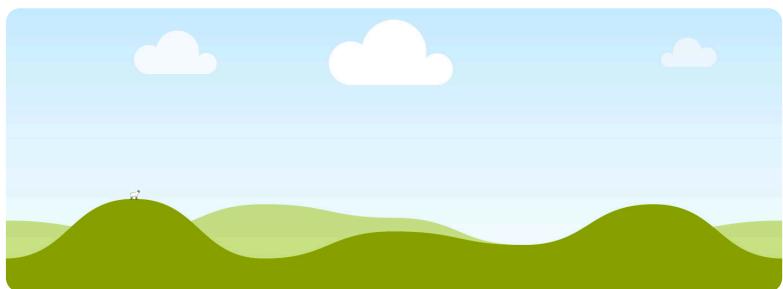
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

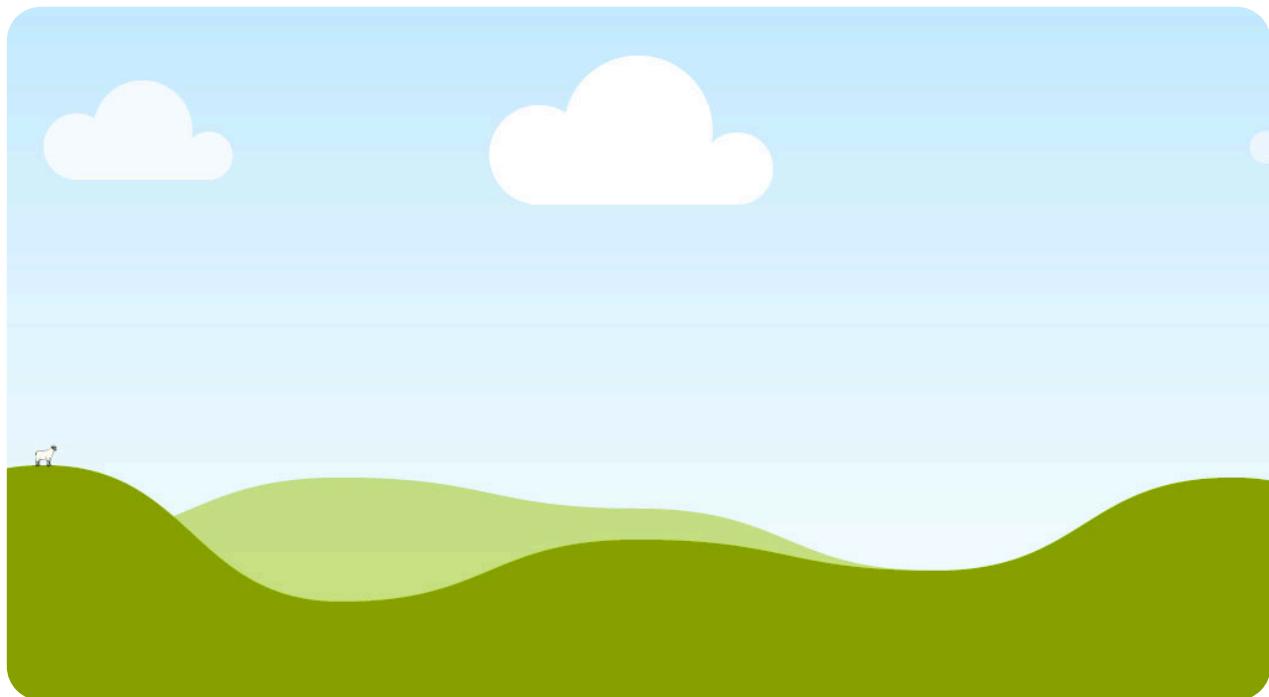


CAPITULO 26: Texto Multilínea y Botones

TEXTO MULTILÍNEA Y BOTONES

SE BASA EN:

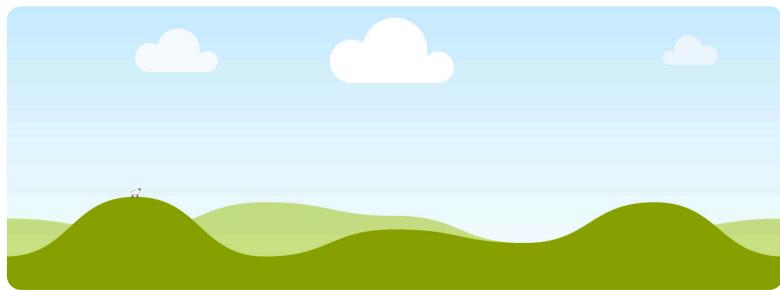
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

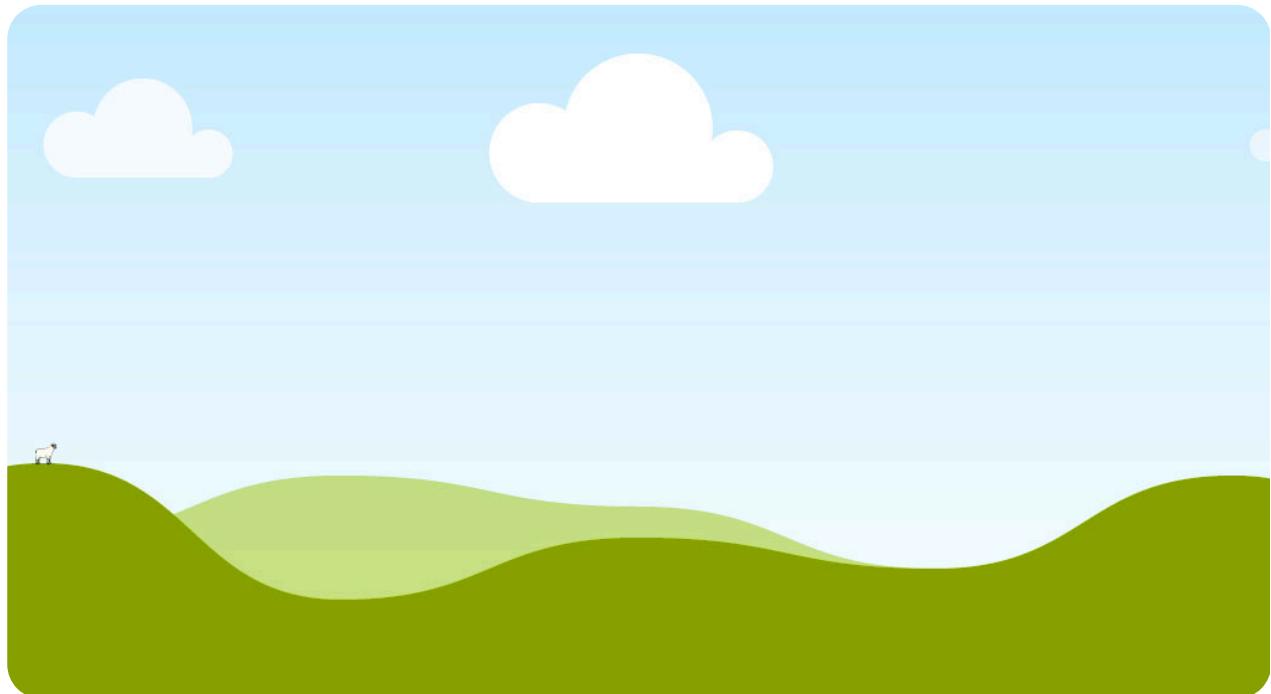


CAPITULO 27: Menús

MENÚS

SE BASA EN:

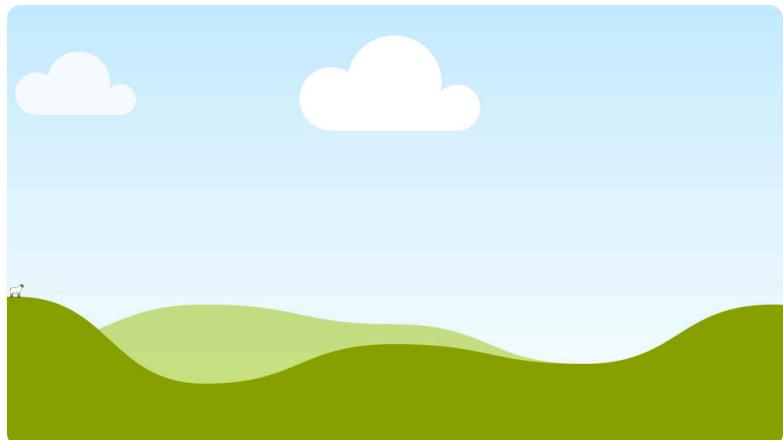
Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

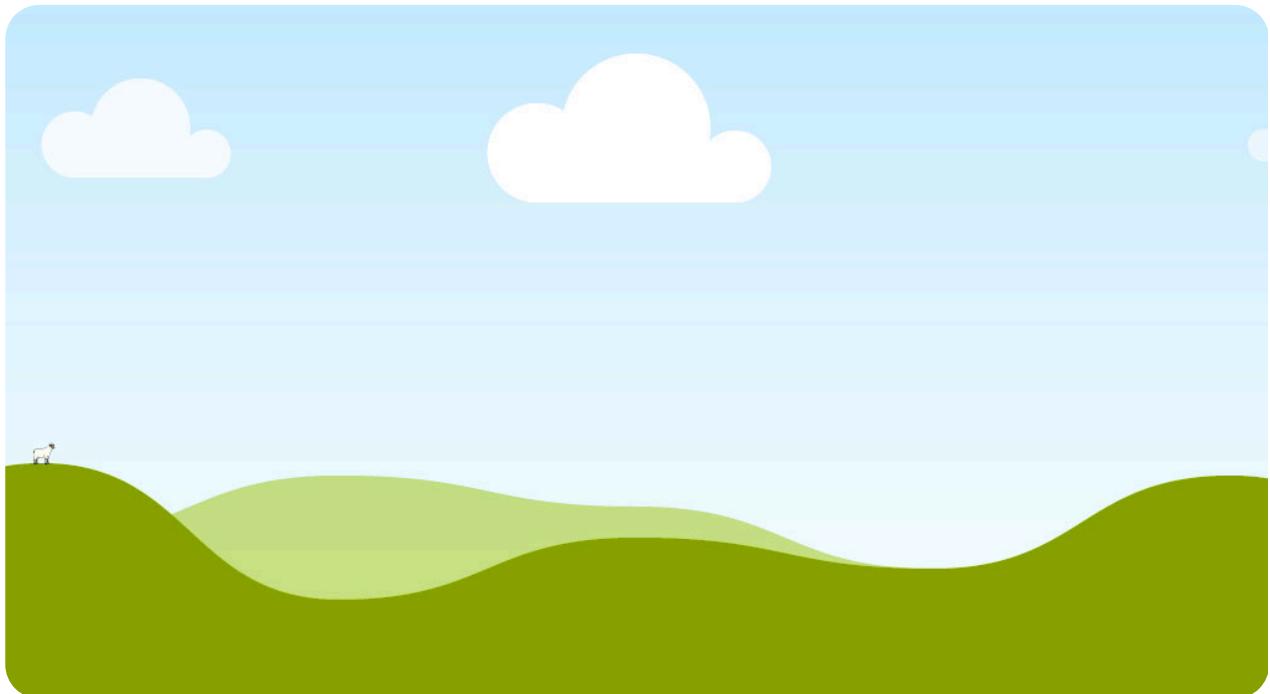


CAPITULO 28: Explorador y Listas

EXPLORADOR Y LISTAS

SE BASA EN:

Un carácter que representa una operación. Un grupo de operadores que se usan en Python son los operadores aritméticos. Estas operaciones pueden ser aritméticas, de comparación, lógicas, de asignación, entre otras.



TIPOS DE

Operadores aritméticos:

Se utilizan para realizar operaciones matemáticas básicas, como suma (+), resta (-), multiplicación (*), división (/), módulo (%), y potencia (**).

