

Python																																																													
<p>Python es un lenguaje de programación de alto nivel conocido por su sintaxis clara y legible. En Python, las variables son contenedores que se utilizan para almacenar datos en la memoria durante la ejecución del programa.</p> <p>Los tipos de variables que podemos encontrar en Python:</p> <ul style="list-style-type: none"><li>• <b>Enteros (int)</b>: Representan números enteros.</li><li>• <b>Flotantes (float)</b>: Representan números con parte decimal.</li><li>• <b>Complejos (complex)</b>: Representan valores numéricos con parte real e imaginaria.</li><li>• <b>Cadenas de caracteres (str)</b>: Representan texto.</li><li>• <b>Booleanos (bool)</b>: Representan valores lógicos Verdadero ('True') o Falso ('False').</li><li>• <b>NoneType (None)</b>: Representa la ausencia de valor o un valor nulo.</li><li>• <b>Fecha y Hora (datetime)</b>: Python proporciona módulos como datetime y time para trabajar con fechas y horas.</li></ul>																																																													
<h3>Funciones básicas</h3> <table border="1"><tr><td><code>print()</code></td><td>Se utiliza para imprimir mensajes, variables y otros objetos.</td></tr><tr><td><code>type()</code></td><td>Nos permite conocer el tipo de datos de una variable o un objeto en Python.</td></tr><tr><td><code>isinstance()</code></td><td>Verifica si un objeto es una instancia de una clase determinada.</td></tr></table>		<code>print()</code>	Se utiliza para imprimir mensajes, variables y otros objetos.	<code>type()</code>	Nos permite conocer el tipo de datos de una variable o un objeto en Python.	<code>isinstance()</code>	Verifica si un objeto es una instancia de una clase determinada.																																																						
<code>print()</code>	Se utiliza para imprimir mensajes, variables y otros objetos.																																																												
<code>type()</code>	Nos permite conocer el tipo de datos de una variable o un objeto en Python.																																																												
<code>isinstance()</code>	Verifica si un objeto es una instancia de una clase determinada.																																																												
<h3>Operaciones aritméticas</h3> <table border="1"><tr><td>+ (Suma)</td><td>* (Multiplicación)</td><td>// (División entera)</td><td>** (Potenciación)</td></tr><tr><td>- (Resta)</td><td>% (Módulo)</td><td>/ (División)</td><td></td></tr></table>		+ (Suma)	* (Multiplicación)	// (División entera)	** (Potenciación)	- (Resta)	% (Módulo)	/ (División)																																																					
+ (Suma)	* (Multiplicación)	// (División entera)	** (Potenciación)																																																										
- (Resta)	% (Módulo)	/ (División)																																																											
<h3>Operaciones binarias</h3> <table border="1"><tr><td>== (Coincidencia)</td><td>!= (Diferencia)</td><td>&gt; (Mayor que)</td><td>&lt; (Menor que)</td><td>and (Se cumplen ambos)</td></tr><tr><td>is (Valores iguales)</td><td>is not (Valores distintos)</td><td>&gt;= (Mayor o igual)</td><td>&lt;= (Menor o igual)</td><td>or (Se cumple uno u otro)</td></tr></table>		== (Coincidencia)	!= (Diferencia)	> (Mayor que)	< (Menor que)	and (Se cumplen ambos)	is (Valores iguales)	is not (Valores distintos)	>= (Mayor o igual)	<= (Menor o igual)	or (Se cumple uno u otro)																																																		
== (Coincidencia)	!= (Diferencia)	> (Mayor que)	< (Menor que)	and (Se cumplen ambos)																																																									
is (Valores iguales)	is not (Valores distintos)	>= (Mayor o igual)	<= (Menor o igual)	or (Se cumple uno u otro)																																																									
<h3>Encadenar texto</h3> <pre>print('Data' + ' ' + 'Science')</pre> <p>Data Science</p>																																																													
<h3>Métodos strings</h3> <table border="1"><tr><td><code>capitalize()</code></td><td>Convierte el primer carácter del string en mayúscula y el resto en minúscula.</td></tr><tr><td><code>count(sub[, start[, end]])</code></td><td>Cuenta cuántas veces aparece una subcadena dentro del string.</td></tr><tr><td><code>find(sub[, start[, end]])</code></td><td>Encuentra la primera aparición de una subcadena en el string.</td></tr><tr><td><code>index(sub[, start[, end]])</code></td><td>Similar a find(), lanza un ValueError si no se encuentra la subcadena.</td></tr><tr><td><code>join(iterable)</code></td><td>Concatena los elementos de un iterable utilizando el string como separador.</td></tr><tr><td><code>lower()</code></td><td>Convierte todos los caracteres del string en minúsculas.</td></tr><tr><td><code>replace(old, new[, count])</code></td><td>Reemplaza todas las ocurrencias de una subcadena con otra subcadena.</td></tr><tr><td><code>split(sep=None, maxsplit=-1)</code></td><td>Divide el string en una lista de subcadenas utilizando un separador.</td></tr><tr><td><code>strip([chars])</code></td><td>Elimina los caracteres especificados del lado izquierdo y derecho del string.</td></tr><tr><td><code>title()</code></td><td>Convierte el string en un título capitalizando la primera letra de cada palabra.</td></tr><tr><td><code>upper()</code></td><td>Convierte todos los caracteres del string en mayúsculas.</td></tr></table>		<code>capitalize()</code>	Convierte el primer carácter del string en mayúscula y el resto en minúscula.	<code>count(sub[, start[, end]])</code>	Cuenta cuántas veces aparece una subcadena dentro del string.	<code>find(sub[, start[, end]])</code>	Encuentra la primera aparición de una subcadena en el string.	<code>index(sub[, start[, end]])</code>	Similar a find(), lanza un ValueError si no se encuentra la subcadena.	<code>join(iterable)</code>	Concatena los elementos de un iterable utilizando el string como separador.	<code>lower()</code>	Convierte todos los caracteres del string en minúsculas.	<code>replace(old, new[, count])</code>	Reemplaza todas las ocurrencias de una subcadena con otra subcadena.	<code>split(sep=None, maxsplit=-1)</code>	Divide el string en una lista de subcadenas utilizando un separador.	<code>strip([chars])</code>	Elimina los caracteres especificados del lado izquierdo y derecho del string.	<code>title()</code>	Convierte el string en un título capitalizando la primera letra de cada palabra.	<code>upper()</code>	Convierte todos los caracteres del string en mayúsculas.																																						
<code>capitalize()</code>	Convierte el primer carácter del string en mayúscula y el resto en minúscula.																																																												
<code>count(sub[, start[, end]])</code>	Cuenta cuántas veces aparece una subcadena dentro del string.																																																												
<code>find(sub[, start[, end]])</code>	Encuentra la primera aparición de una subcadena en el string.																																																												
<code>index(sub[, start[, end]])</code>	Similar a find(), lanza un ValueError si no se encuentra la subcadena.																																																												
<code>join(iterable)</code>	Concatena los elementos de un iterable utilizando el string como separador.																																																												
<code>lower()</code>	Convierte todos los caracteres del string en minúsculas.																																																												
<code>replace(old, new[, count])</code>	Reemplaza todas las ocurrencias de una subcadena con otra subcadena.																																																												
<code>split(sep=None, maxsplit=-1)</code>	Divide el string en una lista de subcadenas utilizando un separador.																																																												
<code>strip([chars])</code>	Elimina los caracteres especificados del lado izquierdo y derecho del string.																																																												
<code>title()</code>	Convierte el string en un título capitalizando la primera letra de cada palabra.																																																												
<code>upper()</code>	Convierte todos los caracteres del string en mayúsculas.																																																												
<h3>Otro métodos strings</h3> <table border="1"><tr><td><code>casefold()</code></td><td>Devuelve una versión en minúsculas del string.</td></tr><tr><td><code>center(width[, fillchar])</code></td><td>Centra el string en un campo de ancho fijo.</td></tr><tr><td><code>endswith(suffix[, start[, end]])</code></td><td>Comprueba si el string termina con el sufijo dado.</td></tr><tr><td><code>expandtabs(tabsize=8)</code></td><td>Expande las tabulaciones en espacios.</td></tr><tr><td><code>find(sub[, start[, end]])</code></td><td>Encuentra la primera aparición de una subcadena en el string.</td></tr><tr><td><code>isalnum()</code></td><td>Comprueba si todos los caracteres del string son alfanuméricos.</td></tr><tr><td><code>isalpha()</code></td><td>Comprueba si todos los caracteres del string son alfabéticos.</td></tr><tr><td><code>isascii()</code></td><td>Comprueba si el string contiene solo caracteres ASCII.</td></tr><tr><td><code>isdecimal()</code></td><td>Comprueba si el string contiene solo caracteres decimales.</td></tr><tr><td><code>isdigit()</code></td><td>Comprueba si el string contiene solo dígitos.</td></tr><tr><td><code>isidentifier()</code></td><td>Comprueba si el string es un identificador válido de Python.</td></tr><tr><td><code>islower()</code></td><td>Comprueba si todos los caracteres del string están en minúsculas.</td></tr><tr><td><code>isnumeric()</code></td><td>Comprueba si el string contiene solo caracteres numéricos.</td></tr><tr><td><code>isprintable()</code></td><td>Comprueba si todos los caracteres del string son imprimibles.</td></tr><tr><td><code>isspace()</code></td><td>Comprueba si el string contiene solo espacios en blanco.</td></tr><tr><td><code>istitle()</code></td><td>Comprueba si el string sigue las reglas de capitalización de un título.</td></tr><tr><td><code>isupper()</code></td><td>Comprueba si todos los caracteres del string están en mayúsculas.</td></tr><tr><td><code>ljust(width[, fillchar])</code></td><td>Justifica el string a la izquierda en un campo de ancho fijo.</td></tr><tr><td><code>lstrip([chars])</code></td><td>Elimina los caracteres especificados del lado izquierdo del string.</td></tr><tr><td><code>partition(sep)</code></td><td>Divide el string en tres partes utilizando el separador especificado.</td></tr><tr><td><code>rfind(sub[, start[, end]])</code></td><td>Similar a find(), pero busca desde el final del string.</td></tr><tr><td><code>rindex(sub[, start[, end]])</code></td><td>Similar a index(), pero busca desde el final del string.</td></tr><tr><td><code>rjust(width[, fillchar])</code></td><td>Justifica el string a la derecha en un campo de ancho fijo.</td></tr><tr><td><code>rsplit(sep=None, maxsplit=-1)</code></td><td>Divide el string en una lista de subcadenas, comenzando desde el final.</td></tr><tr><td><code>rstrip([chars])</code></td><td>Elimina los caracteres especificados del lado derecho del string.</td></tr><tr><td><code>splittines([keepends])</code></td><td>Divide el string en una lista de líneas.</td></tr><tr><td><code>startswith(prefix[, start[, end]])</code></td><td>Comprueba si el string comienza con el prefijo dado.</td></tr><tr><td><code>swapcase()</code></td><td>Intercambia entre mayúsculas y minúsculas en todo el string.</td></tr><tr><td><code>translate(table)</code></td><td>Aplica una tabla de traducción a cada carácter en el string.</td></tr><tr><td><code>zfill(width)</code></td><td>Rellena el string con '0' a la izquierda hasta alcanzar la longitud especificada.</td></tr></table>		<code>casefold()</code>	Devuelve una versión en minúsculas del string.	<code>center(width[, fillchar])</code>	Centra el string en un campo de ancho fijo.	<code>endswith(suffix[, start[, end]])</code>	Comprueba si el string termina con el sufijo dado.	<code>expandtabs(tabsize=8)</code>	Expande las tabulaciones en espacios.	<code>find(sub[, start[, end]])</code>	Encuentra la primera aparición de una subcadena en el string.	<code>isalnum()</code>	Comprueba si todos los caracteres del string son alfanuméricos.	<code>isalpha()</code>	Comprueba si todos los caracteres del string son alfabéticos.	<code>isascii()</code>	Comprueba si el string contiene solo caracteres ASCII.	<code>isdecimal()</code>	Comprueba si el string contiene solo caracteres decimales.	<code>isdigit()</code>	Comprueba si el string contiene solo dígitos.	<code>isidentifier()</code>	Comprueba si el string es un identificador válido de Python.	<code>islower()</code>	Comprueba si todos los caracteres del string están en minúsculas.	<code>isnumeric()</code>	Comprueba si el string contiene solo caracteres numéricos.	<code>isprintable()</code>	Comprueba si todos los caracteres del string son imprimibles.	<code>isspace()</code>	Comprueba si el string contiene solo espacios en blanco.	<code>istitle()</code>	Comprueba si el string sigue las reglas de capitalización de un título.	<code>isupper()</code>	Comprueba si todos los caracteres del string están en mayúsculas.	<code>ljust(width[, fillchar])</code>	Justifica el string a la izquierda en un campo de ancho fijo.	<code>lstrip([chars])</code>	Elimina los caracteres especificados del lado izquierdo del string.	<code>partition(sep)</code>	Divide el string en tres partes utilizando el separador especificado.	<code>rfind(sub[, start[, end]])</code>	Similar a find(), pero busca desde el final del string.	<code>rindex(sub[, start[, end]])</code>	Similar a index(), pero busca desde el final del string.	<code>rjust(width[, fillchar])</code>	Justifica el string a la derecha en un campo de ancho fijo.	<code>rsplit(sep=None, maxsplit=-1)</code>	Divide el string en una lista de subcadenas, comenzando desde el final.	<code>rstrip([chars])</code>	Elimina los caracteres especificados del lado derecho del string.	<code>splittines([keepends])</code>	Divide el string en una lista de líneas.	<code>startswith(prefix[, start[, end]])</code>	Comprueba si el string comienza con el prefijo dado.	<code>swapcase()</code>	Intercambia entre mayúsculas y minúsculas en todo el string.	<code>translate(table)</code>	Aplica una tabla de traducción a cada carácter en el string.	<code>zfill(width)</code>	Rellena el string con '0' a la izquierda hasta alcanzar la longitud especificada.
<code>casefold()</code>	Devuelve una versión en minúsculas del string.																																																												
<code>center(width[, fillchar])</code>	Centra el string en un campo de ancho fijo.																																																												
<code>endswith(suffix[, start[, end]])</code>	Comprueba si el string termina con el sufijo dado.																																																												
<code>expandtabs(tabsize=8)</code>	Expande las tabulaciones en espacios.																																																												
<code>find(sub[, start[, end]])</code>	Encuentra la primera aparición de una subcadena en el string.																																																												
<code>isalnum()</code>	Comprueba si todos los caracteres del string son alfanuméricos.																																																												
<code>isalpha()</code>	Comprueba si todos los caracteres del string son alfabéticos.																																																												
<code>isascii()</code>	Comprueba si el string contiene solo caracteres ASCII.																																																												
<code>isdecimal()</code>	Comprueba si el string contiene solo caracteres decimales.																																																												
<code>isdigit()</code>	Comprueba si el string contiene solo dígitos.																																																												
<code>isidentifier()</code>	Comprueba si el string es un identificador válido de Python.																																																												
<code>islower()</code>	Comprueba si todos los caracteres del string están en minúsculas.																																																												
<code>isnumeric()</code>	Comprueba si el string contiene solo caracteres numéricos.																																																												
<code>isprintable()</code>	Comprueba si todos los caracteres del string son imprimibles.																																																												
<code>isspace()</code>	Comprueba si el string contiene solo espacios en blanco.																																																												
<code>istitle()</code>	Comprueba si el string sigue las reglas de capitalización de un título.																																																												
<code>isupper()</code>	Comprueba si todos los caracteres del string están en mayúsculas.																																																												
<code>ljust(width[, fillchar])</code>	Justifica el string a la izquierda en un campo de ancho fijo.																																																												
<code>lstrip([chars])</code>	Elimina los caracteres especificados del lado izquierdo del string.																																																												
<code>partition(sep)</code>	Divide el string en tres partes utilizando el separador especificado.																																																												
<code>rfind(sub[, start[, end]])</code>	Similar a find(), pero busca desde el final del string.																																																												
<code>rindex(sub[, start[, end]])</code>	Similar a index(), pero busca desde el final del string.																																																												
<code>rjust(width[, fillchar])</code>	Justifica el string a la derecha en un campo de ancho fijo.																																																												
<code>rsplit(sep=None, maxsplit=-1)</code>	Divide el string en una lista de subcadenas, comenzando desde el final.																																																												
<code>rstrip([chars])</code>	Elimina los caracteres especificados del lado derecho del string.																																																												
<code>splittines([keepends])</code>	Divide el string en una lista de líneas.																																																												
<code>startswith(prefix[, start[, end]])</code>	Comprueba si el string comienza con el prefijo dado.																																																												
<code>swapcase()</code>	Intercambia entre mayúsculas y minúsculas en todo el string.																																																												
<code>translate(table)</code>	Aplica una tabla de traducción a cada carácter en el string.																																																												
<code>zfill(width)</code>	Rellena el string con '0' a la izquierda hasta alcanzar la longitud especificada.																																																												
<h3>Listas [ valor_1, valor_2, ..., valor_n ]</h3> <table border="1"><tr><td><code>lista = [valor_1, valor_2, valor_3]</code></td><td>Crea una lista de elementos.</td></tr><tr><td><code>lista = []</code></td><td>Crea una lista vacía.</td></tr><tr><td><code>append()</code></td><td>Agrega un elemento al final de la lista.</td></tr><tr><td><code>extend()</code></td><td>Agrega varios elementos al final de la lista.</td></tr><tr><td><code>insert()</code></td><td>Agrega un elemento en una posición específica de la lista.</td></tr><tr><td><code>remove()</code></td><td>Elimina la primera aparición de un elemento de la lista.</td></tr><tr><td><code>pop()</code></td><td>Elimina y devuelve el elemento en una posición específica de la lista.</td></tr><tr><td><code>clear()</code></td><td>Elimina todos los elementos de la lista.</td></tr><tr><td><code>index()</code></td><td>Devuelve la posición de la primera aparición de un elemento en la lista.</td></tr><tr><td><code>count()</code></td><td>Devuelve el número de veces que aparece un elemento en la lista.</td></tr><tr><td><code>sort()</code></td><td>Ordena los elementos de la lista.</td></tr><tr><td><code>reverse()</code></td><td>Invierte el orden de los elementos de la lista.</td></tr><tr><td><code>copy()</code></td><td>Hace una copia exacta de una lista con los mismos elementos que la original.</td></tr></table>		<code>lista = [valor_1, valor_2, valor_3]</code>	Crea una lista de elementos.	<code>lista = []</code>	Crea una lista vacía.	<code>append()</code>	Agrega un elemento al final de la lista.	<code>extend()</code>	Agrega varios elementos al final de la lista.	<code>insert()</code>	Agrega un elemento en una posición específica de la lista.	<code>remove()</code>	Elimina la primera aparición de un elemento de la lista.	<code>pop()</code>	Elimina y devuelve el elemento en una posición específica de la lista.	<code>clear()</code>	Elimina todos los elementos de la lista.	<code>index()</code>	Devuelve la posición de la primera aparición de un elemento en la lista.	<code>count()</code>	Devuelve el número de veces que aparece un elemento en la lista.	<code>sort()</code>	Ordena los elementos de la lista.	<code>reverse()</code>	Invierte el orden de los elementos de la lista.	<code>copy()</code>	Hace una copia exacta de una lista con los mismos elementos que la original.																																		
<code>lista = [valor_1, valor_2, valor_3]</code>	Crea una lista de elementos.																																																												
<code>lista = []</code>	Crea una lista vacía.																																																												
<code>append()</code>	Agrega un elemento al final de la lista.																																																												
<code>extend()</code>	Agrega varios elementos al final de la lista.																																																												
<code>insert()</code>	Agrega un elemento en una posición específica de la lista.																																																												
<code>remove()</code>	Elimina la primera aparición de un elemento de la lista.																																																												
<code>pop()</code>	Elimina y devuelve el elemento en una posición específica de la lista.																																																												
<code>clear()</code>	Elimina todos los elementos de la lista.																																																												
<code>index()</code>	Devuelve la posición de la primera aparición de un elemento en la lista.																																																												
<code>count()</code>	Devuelve el número de veces que aparece un elemento en la lista.																																																												
<code>sort()</code>	Ordena los elementos de la lista.																																																												
<code>reverse()</code>	Invierte el orden de los elementos de la lista.																																																												
<code>copy()</code>	Hace una copia exacta de una lista con los mismos elementos que la original.																																																												
<h3>Operaciones binarias</h3> <table border="1"><tr><td><code>len()</code> (Nº elementos de la lista)</td><td><code>max()</code> (Elemento máximo de la lista)</td><td><code>min()</code> (Elemento mínimo de la lista)</td></tr></table>		<code>len()</code> (Nº elementos de la lista)	<code>max()</code> (Elemento máximo de la lista)	<code>min()</code> (Elemento mínimo de la lista)																																																									
<code>len()</code> (Nº elementos de la lista)	<code>max()</code> (Elemento máximo de la lista)	<code>min()</code> (Elemento mínimo de la lista)																																																											
<h3>Indexación de listas</h3> <table border="1"><tr><td><code>lista[i]</code></td><td>Devuelve el elemento en la posición i.</td></tr><tr><td><code>lista[i:j]</code></td><td>Devuelve los elementos entre la posición i y la j (esta última sin incluir).</td></tr><tr><td><code>lista[i:]</code></td><td>Devuelve los elementos entre la posición i y el final de la lista.</td></tr><tr><td><code>lista[:j]</code></td><td>Devuelve los elementos hasta la posición j (esta última sin incluir).</td></tr><tr><td><code>lista[i:j:k]</code></td><td>Devuelve los elementos entre la posición i y la j saltando k elementos.</td></tr></table>		<code>lista[i]</code>	Devuelve el elemento en la posición i.	<code>lista[i:j]</code>	Devuelve los elementos entre la posición i y la j (esta última sin incluir).	<code>lista[i:]</code>	Devuelve los elementos entre la posición i y el final de la lista.	<code>lista[:j]</code>	Devuelve los elementos hasta la posición j (esta última sin incluir).	<code>lista[i:j:k]</code>	Devuelve los elementos entre la posición i y la j saltando k elementos.																																																		
<code>lista[i]</code>	Devuelve el elemento en la posición i.																																																												
<code>lista[i:j]</code>	Devuelve los elementos entre la posición i y la j (esta última sin incluir).																																																												
<code>lista[i:]</code>	Devuelve los elementos entre la posición i y el final de la lista.																																																												
<code>lista[:j]</code>	Devuelve los elementos hasta la posición j (esta última sin incluir).																																																												
<code>lista[i:j:k]</code>	Devuelve los elementos entre la posición i y la j saltando k elementos.																																																												
<h3>Tuplas ( valor_1, valor_2, ..., valor_n )</h3> <table border="1"><tr><td><code>tupla = (valor_1, valor_2, valor_3)</code></td><td>Crea una tupla de elementos.</td></tr><tr><td><code>tupla = ()</code></td><td>Crea una tupla vacía</td></tr><tr><td><code>index()</code></td><td>Devuelve el índice de la primera ocurrencia de un elemento en la tupla.</td></tr><tr><td><code>count()</code></td><td>Devuelve el número de veces que un elemento aparece en la tupla.</td></tr></table>		<code>tupla = (valor_1, valor_2, valor_3)</code>	Crea una tupla de elementos.	<code>tupla = ()</code>	Crea una tupla vacía	<code>index()</code>	Devuelve el índice de la primera ocurrencia de un elemento en la tupla.	<code>count()</code>	Devuelve el número de veces que un elemento aparece en la tupla.																																																				
<code>tupla = (valor_1, valor_2, valor_3)</code>	Crea una tupla de elementos.																																																												
<code>tupla = ()</code>	Crea una tupla vacía																																																												
<code>index()</code>	Devuelve el índice de la primera ocurrencia de un elemento en la tupla.																																																												
<code>count()</code>	Devuelve el número de veces que un elemento aparece en la tupla.																																																												
<h3>Propiedades Tuplas</h3> <table border="1"><tr><td><code>len()</code> (Nº elementos de la tupla)</td><td><code>max()</code> (Elemento máximo de la tupla)</td><td><code>min()</code> (Elemento mínimo de la tupla)</td></tr></table>		<code>len()</code> (Nº elementos de la tupla)	<code>max()</code> (Elemento máximo de la tupla)	<code>min()</code> (Elemento mínimo de la tupla)																																																									
<code>len()</code> (Nº elementos de la tupla)	<code>max()</code> (Elemento máximo de la tupla)	<code>min()</code> (Elemento mínimo de la tupla)																																																											
<h3>Indexación de tuplas</h3> <table border="1"><tr><td><code>tupla[i]</code></td><td>Devuelve el elemento en la posición i.</td></tr><tr><td><code>tupla[i:j]</code></td><td>Devuelve los elementos entre la posición i y la j (esta última sin incluir).</td></tr><tr><td><code>tupla[i:]</code></td><td>Devuelve los elementos entre la posición i y el final de la lista.</td></tr><tr><td><code>tupla[:j]</code></td><td>Devuelve los elementos hasta la posición j (esta última sin incluir).</td></tr><tr><td><code>tupla[i:j:k]</code></td><td>Devuelve los elementos entre la posición i y la j saltando k elementos.</td></tr></table>		<code>tupla[i]</code>	Devuelve el elemento en la posición i.	<code>tupla[i:j]</code>	Devuelve los elementos entre la posición i y la j (esta última sin incluir).	<code>tupla[i:]</code>	Devuelve los elementos entre la posición i y el final de la lista.	<code>tupla[:j]</code>	Devuelve los elementos hasta la posición j (esta última sin incluir).	<code>tupla[i:j:k]</code>	Devuelve los elementos entre la posición i y la j saltando k elementos.																																																		
<code>tupla[i]</code>	Devuelve el elemento en la posición i.																																																												
<code>tupla[i:j]</code>	Devuelve los elementos entre la posición i y la j (esta última sin incluir).																																																												
<code>tupla[i:]</code>	Devuelve los elementos entre la posición i y el final de la lista.																																																												
<code>tupla[:j]</code>	Devuelve los elementos hasta la posición j (esta última sin incluir).																																																												
<code>tupla[i:j:k]</code>	Devuelve los elementos entre la posición i y la j saltando k elementos.																																																												
<h3>Función Zip()</h3> <p>La función <code>zip()</code> crea pares de elementos, combinando el primer elemento de cada iterable, luego el segundo elemento de cada uno, y así sucesivamente.</p> <pre>iterable_1 = [elemento_1, elemento_2, elemento_3] iterable_2 = [elemento_a, elemento_b, elemento_c]  lista = list(zip(iterable_1, iterable_2))  print(lista) #Output: [(elemento_1, elemento_a), (elemento_2, elemento_b), (elemento_3, elemento_c),</pre>																																																													

Diccionarios {clave_1: valor_1, clave_2: valor_2, ..., clave_n: valor_n}	
<code>dicc = {}</code>	Crea un diccionario vacío.
<code>dicc = {clave_1:valor_1, clave_2:valor_2}</code>	Crea un diccionario de pares clave-valor.
<code>dicc=dict(cl_1 = vlr_1, cl_2=vlr_2)</code>	Crea un diccionario usando el constructor dict().
<code>dicc = dict(lista_de_tuplas)</code>	Crea un diccionario a partir de una lista de tuplas.
<code>dicc = dict.fromkeys(claves, valor)</code>	Crea un diccionario con único valor para todas las claves.
Propiedades Diccionarios	
<code>len()</code> (Nºelementos del diccionario)	<code>keys()</code> (Vista con todas las claves del diccionario)
<code>values()</code> (Vista con todos los valores del diccionario)	<code>items()</code> (Vista con pares (clave,valor))
Métodos Diccionarios	
<code>copy()</code>	Devuelve una copia superficial del diccionario.
<code>clear()</code>	Elimina todos los elementos del diccionario.
<code>update()</code>	Actualiza con pares clave-valor de otro dicc o de una lista de tuplas.
<code>get(x,y)</code>	Devuelve el valor de la clave x si está presente en el diccionario, sino y.
<code>setdefault(x,y)</code>	Si x está en el diccionario, devuelve su valor. Si no, la inserta con el valor y.
<code>pop()</code>	Elimina la clave y devuelve su valor
<code>popitem()</code>	Elimina y devuelve un par clave-valor arbitrario del diccionario
<code>sorted()</code>	Devuelve una lista con las claves del diccionario ordenadas.
<code>strip([chars])</code>	Elimina los caracteres especificados del lado izquierdo y derecho del string.
<code>title()</code>	Convierte el string en un título capitalizando la primera letra de cada palabra.
<code>upper()</code>	Convierte todos los caracteres del string en mayúsculas.
Sets {clave_1, clave_2, ..., clave_n}	
<code>sets = {clave_1, clave_2, clave_3}</code>	Crea un set de elementos.
<code>sets = set(clave_1, clave_2, clave_3)</code>	Crea un set usando el constructor set()
Propiedades Sets	
<code>len()</code> (Nºelementos del set)	<code>min()</code> (Elemento más pequeño del set)
<code>max()</code> (Elemento más grande del set)	<code>in y not in</code> (Verifican si un elemento está o no en el set)
Métodos Sets	
<code>add()</code>	Agrega un elemento al conjunto. Si ya está presente, no se hace nada.
<code>update()</code>	Agrega múltiples elementos al conjunto.
<code>copy()</code>	Crea una copia superficial del conjunto existente.
<code>pop()</code>	Elimina y devuelve un elemento aleatorio del set.
<code>remove()</code>	Elimina un elemento del conjunto.
<code>discard()</code>	Elimina un elemento del conjunto, sin error si el elemento no está presente.
<code>clear()</code>	Elimina todos los elementos del conjunto.

Operaciones de conjuntos con Sets	
<code>union()</code>	Crea un nuevo conjunto con todos los elementos únicos de ambos conjuntos.
<code>intersection()</code>	Produce un conjunto que solo tiene elementos presentes en ambos conjuntos.
<code>difference()</code>	Crea un set con elementos del primer conjunto que no están en el segundo.
<code>symmetric_difference()</code>	Crea un set que contiene elementos que no están en ambos conjuntos.
<code>isdisjoint()</code>	Se utiliza para verificar si dos conjuntos no tienen elementos en común.
<code>issubset()</code>	Determina si un conjunto es un subconjunto de otro.
<code>issuperset()</code>	Determina si un conjunto contiene a otro conjunto.
Estructuras de selección	
<code>if</code> nota > 5: print('Aprobado') <code>elif</code> nota < 5: print('Suspensos') <code>else:</code> print('Suficiente')	La función <code>if</code> se utiliza para ejecutar un bloque de código solo si una condición específica se evalúa como verdadera. La función <code>elif</code> se utiliza para evaluar una condición adicional si la condición del "if" anterior fue falsa. Puede haber tantas funciones <code>elif</code> para evaluar condiciones como sea necesario. La función <code>else</code> se utiliza para ejecutar un bloque de código si ninguna de las condiciones anteriores son verdaderas.
Estructuras de iteración. Bucle While	
<code>x = 5 while x &gt;0:     x -= 1     print('El valor de x es mayor de 0') print('El valor de x es 0')</code>	El bucle <code>while</code> permite repetir un bloque de código mientras una condición dada se cumpla. Se utiliza cuando no sabemos cuántas veces necesitaremos ejecutar el código.
<code>x = 5 while x &gt;0:     print('El valor de x es mayor de 0') print('El valor de x es 0')</code>	Si la condición nunca llega a ser falsa se crearía un bucle infinito.
Estructuras de iteración. Bucle For	
<code>for key in dictionary:     # hacer algo con la clave o el valor</code>	El bucle <code>for</code> se utiliza para iterar sobre una secuencia de elementos, como una lista, una tupla, un diccionario o un set, y realizar una acción en cada uno de ellos.

List comprehensions	
Las <b>list comprehensions</b> son una forma concisa y poderosa de crear listas. Permiten generar listas de manera eficiente a partir de iterables como listas, tuplas, o rangos, aplicando transformaciones o filtros a cada elemento en un solo paso.	
<code>nueva_lista = [expresion for elemento in iterable]</code>	
<code>List Comprehensions con if</code>	<code>lista = [expresion for elemento in iterable if condicion]</code>
<code>List Comprehensions con if y else</code>	<code>lista = [expresion_true if condicion else expresion_false for elemento in iterable]</code>
<code>Nested List Comprehensions</code>	<code>matriz = [expresion for elemento in fila for fila in matriz_original]</code>
<code>symmetric_difference()</code>	Crea un set que contiene elementos que no están en ambos conjuntos.
<code>isdisjoint()</code>	Se utiliza para verificar si dos conjuntos no tienen elementos en común
<code>issubset()</code>	Determina si un conjunto es un subconjunto de otro
<code>issuperset()</code>	Determina si un conjunto contiene a otro conjunto
Funciones	

Las <b>funciones</b> en Python son bloques de código que se invocan para realizar tareas específicas. Su propósito es organizar el código en segmentos más manejables, lo que facilita su comprensión, lectura y mantenimiento.
<code>def nombre_función (parámetro_1, parámetro_2, ..., parámetro_n):     #operaciones a realizar     return valor_de_salida.</code>
Para llamar a la función seguiremos la sintaxis general:
<code>nombre_función (argumento_1, argumento_2)</code>
*args: Permiten pasar un número variable de argumentos posicionales a una función. **kwargs: Permite pasar un número variable de argumentos clave-valor a una función.
La <b>recursividad</b> consiste en que una función se llame a sí misma durante su propia ejecución. Esto permite que la función se repita o resuelva un problema más pequeño dentro de sí misma hasta que se cumpla una condición de terminación, conocida como caso base.
<code>def recursiva (parámetro):     ...     recursiva ()</code>