Índice general

1 Hola mundo

Comillas dobles y simples

Imprime tu nombre utilizando la función `print()`. Lo tienes que escribir con comillas dobles y con comillas simples.

Dos líneas con dos sentencias

Imprime tu nombre y tus apellidos en dos líneas, utilizando dos sentencias `print()`.

Dos líneas con una única sentencia

Imprime tu nombre y tus apellidos en dos líneas, utilizando una única sentencia `print()` y la secuencia de escape `\n`.

† Tabulador

El caracter `\t` es un caracter de escape que imprime una tabulación. Utilizando este caracter imprime la tabla:

Matemáticas	SB
Lengua	SB
Python	MH

Comillas dentro del texto

¿Cómo se puede escribir una comilla simple dentro de un texto? Decir al menos dos formas de hacerlo.

† Repetición de la misma cadena

Prueba los siguientes códigos para entender su funcionamiento:

```
print("hola" * 10)
print("Yo no me llamo Javier"\n * 15)
```

Escribe mil veces la frase:

Debo estudiar Python

2 Enteros I

Sumas

Realiza las siguientes sumas:

$$a)\ 56+890 \qquad b)\ 456+2345+123$$

Restas

Realiza las siguientes restas:

$$a)\ 456-234 \qquad b)\ 345-82612 \qquad c)\ -456-234$$

Multiplicaciones

Realiza las siguientes multiplicaciones:

$$a)\ 45\cdot 123456789 \qquad b)\ (-3)\cdot 67 \qquad c)\ (-34)\cdot(-2)$$

Divisiones

Realiza las siguientes divisiones:

$$a)\ 456/34 \qquad b)\ 56/7 \qquad c)\ (-345)/98765$$

Operaciones combinadas

Realiza las siguientes operaciones combinadas:

$$a)\ 5-8\cdot 67+7-34\cdot 23 \qquad b)\ (2+5-3)(3-78-9)$$

Operaciones combinadas

Intenta adivinar el resultado de las siguientes operaciones antes de realizarlas:

$$a)\, 4+8\cdot 4 \qquad b)\, 5\cdot(-4)+5\cdot 7 \qquad c)\, 8-6/2+1\cdot 2$$

Colocación de paréntesis

Pon paréntesis en la siguiente expresión para obtener tres resultados distintos:

$$4-9*8+7$$

Problema

Tenemos 2 cajones de manzanas con 20 manzanas cada uno. Nos traen otros tres cajones de manzanas con 20 manzanas cada uno. ¿Cuántas manzanas tendremos? Se tiene que resolver con una única operación combinada.

Problema

Hay que repartir una caja de bombones de 24 bombones entre 2 grupos de 3 alumnos cada uno. ¿A cuántos bombones toca cada alumno? Se tiene que resolver con una única operación combinada.

3 Enteros II

Potencias

Calcular las siguientes potencias:

$$a)\, 3^4 \qquad b)\, (-3)^{23} \qquad c)\, 4^{500}$$

Prioridad de las potencias

Calcula el siguiente resultado de cabeza y después comprueba con Python:

$$5 * 2 * *3$$

Colocación de paréntesis

Pon paréntesis en la siguiente operación para obtener tres resultados distintos:

$$3 * 2 * *5 * 5$$

Cociente y resto

Calcula el cociente y el resto de las divisiones:

$$a)\, 4678/23 \qquad b)\, 1234567/9875$$

Cociente y resto

Calcula el cociente y el resto y realiza la comprobación de la división:

$$5635/258$$

Espacios en blanco entre operadores

Comprueba que puedes escribir todos los espacios en blanco que quieras entre los operadores y el número. También se puede no dejar ningún espacio.

† Prioridad de operaciones con //

Haz pruebas para comprobar si se realiza antes el operador // o el operador +.

† División entre cero

Comprueba que si dividimos entre cero obtenemos un error. Depués de esto intenta poner paréntesis en la siguiente operación para obtener un error:

$$4/20-5*4$$

† La función *divmod()*

La función `divmod()` permite calcular el cociente y el resto. Por ejemplo si tecleamos:

```
divmod(43, 5)
```

el resultado es una pareja de números. El primero es cociente y el segundo el resto. Calcula, utilizando `divmod()` el cociente y el resto de la división:

$$456/34$$

† La función *pow()*

La función `pow()` permite calcular potencias. Por ejemplo, prueba los siguientes códigos para entender su funcionamiento:

```
pow(2,4)
pow(2,5)
```

Utilizando la función `pow()` calcula la potencia:

$$(-3)^4$$

† Convertir números a binario

Con la función `bin()` podemos convertir números enteros en su representación en binario. Observar que aparece entre comillas y delante del número también aparece `0b`. El número binario es la parte formada por ceros y unos.

```
bin(45) # Resultado '0b101101'
bin(63) # Resultado '0b111111'
```

Convierte algunos números en binario.

† Convertir de binario a entero

Esta conversión es más complicada. Primero tenemos que escribir el número binario entre comillas y precedido de `0b`. Después debemos emplear la función `int()` de un modo especial, como en el siguiente código:

```
int('0b101101', 2) # Resultado 45
int('0b111111', 2) # Resultado 63
```

4 Números reales

Operaciones combinadas

Realiza las siguientes operaciones:

$$a)\ 45{,}89 \cdot 45{,}7^4 \qquad b)\ 1/3 \cdot 3 + 9$$

Notación científica

Realiza las siguientes operaciones en notación científica:

$$1{,}6 \cdot 10^4 \cdot 4{,}789 \cdot 10^6$$

Convertir pies en metros

Convierte 34,6 pies en metros. Buscar en Internet la fórmula de conversión.

Muchos decimales es imposible

Escribir un número con más de 20 decimales. Comprobar que Python recorta el número, pues no puede almacenar tantos decimales.

La fórmula de Einstein

Calcula la energia liberada por una masa de 5 gramos utilizando la fórmula de Einstein

$$E = mc^2$$

(Buscar en Internet la velocidad de la luz)

† La función *round()*

La función 'round()' sirve para redondear números. Prueba los siguentes códigos para entender el funcionamiento:

```
round(3.14)
round(3.7)
round(-3.7)
```

Notación científica

El número $3{,}67 \cdot 10^5$ se escribe en Python como $3{,}67e5$. Escribe dicho número en forma de *notación científica* y comprueba que coincide con la multiplicación anterior. Comprueba lo mismo con otros números.

Problema

En una hamburguesería, cada hamburguesa cuesta 5.99?, las patatas fritas 1.99? y cada refresco 1.49?. Si vas solo y tomas una hamburguesa con patatas y refresco, ¿cuánto pagarás? Si invitas a tus amigos y se toman en total: 7 hamburguesas, 10 de patatas fritas y 8 refrescos, ¿cuánto te devuelven si pagas con un billete de 100??

5 Variables

Guardar valores en variables

Guarda en una variable llamada `pi` el valor 3.1415.

Perímetro de una circunferencia

La fórmula para calcular el perímetro de una circunferencia es

$$2\pi r$$

Utiliza la variable anterior para calcular el perímetro de una circunferencia de radio 6 m.

Palabras reservadas

Busca en Internet más palabras reservadas de Python y comprueba que no se pueden usar como nombres del variables.

Reasignación de valores a variables

Averigua que resultado imprime el siguiente código, explicando la razón.

```
hola = 90
Hola = 89
print(Hola)
Hola = 120
print(f"La variable contiene el valor {hola}")
```

Nombres permitidos para variables

Algunos de los siguientes nombres de variables están permitidos y otros no son válidos. Explica las razones y comprueba los resultados.

```
a1
velocidad
velocidad99
salario_medio
salario medio
_b
```

Reasignación de valores a variables

Haz un esquema donde se vean los valores que van tomando las variables y adivinar el resultado del siguiente código:

```
deuda = 0
compra = 100
deuda = deuda +  compra
compra = 200
deuda = deuda + compra
print(deuda)
```

† Las letras acentuadas en las variables

¿Se pueden escribir nombres de variables con letras acentuadas? ¿Son distintas las variables acentuadas de otras que se escriben igual pero sin acentuar?

† Borrado de variables con 'del'

Una vez creada una variable sigue existiendo en memoria hasta la terminación del programa. Si queremos borrar una variable utilizamos la orden `del`. Prueba el siguiente código que crea una variable, la imprime y después la borra. Comprueba que la variable ya no existe.

```
a = 10
print(a)
del a
```

† Operaciones en la parte derecha de una asignación

Si queremos asignar a una variable el valor de una cierta operación debemos tener en cuenta que primero se realizan todas las operaciones a la derecha del operador = y después se asigna a la variable. Que valor tendrá `x` al final del programa en el siguiente código:

```
x = 4 + 8 * 2
x = x + 20
x = 3 * x +10
```

[⇑] Inicializar varias variables al mismo tiempo

En general usamos una línea para inicializar cada una de las variables. Sin embargo se pueden inicializar varias a la vez. Para ello debemos escribir los nombres de las variables separados por comas, después el signo de asignación y después los valores que queremos asignar a cada variable, también separados por comas:

```
x, y, z = 6, 7, 8
```

En la primera variable se guarda el primer valor, en la segunda el segundo valor, ... Guarda tu nombre y tu apellido en dos variables utilizando únicamente una línea.

† Operadores en asignación

Una operación muy común en Python es sumar a una variable un cierto valor y volver a guardar el nuevo valor en la misma variable. Por ejemplo

```
x = 6
x = x + 3
```

guarda el valor 9 en la variable x. Como esta operación es tan común, Python ha creado un atajo. Las líneas anteriores de código se podrían haber escrito:

```
x = 6
x += 2
```

El operador += se llama **suma en asignación**. Del mismo modo existe la resta en asignación:

```
x = 45
x -= 4
```

Ahora hemos guardado un 41 en la variable x.
En la siguiente tabla se presentan todos los operadores en asignación y se escribe su equivalencia utilizando el operador =.

Operador	Ejemplo	Equivale a:
+=	x += a	x = x + a
-=	x -= a	x = x - a
*=	x *= a	x = x * a
/=	x /= a	x = x / a
//=	x //= a	x = x // a
%=	x%= a	x = x% a
**=	x **= a	x = x ** a

Calcula el valor final de x en el siguiente código:

```
x = 6
x += 5
x -= 7
x *= 4
```

† Controlar los decimales

Cuando Python trabaja con decimales y los tiene que mostrar por pantalla con la orden `print()` siempre muestra todos los decimales. Además de esto, como Python a veces se equivoca al hacer cuentas, esto provoca un exceso de decimales, como en el código:

```
x = 0.1
y = 0.2
print(f"La suma es {x + y}")
```

Si queremos controlar el número de decimales que presentamos por pantalla, debemos emplear un pequeño «truco». Prueba los siguientes códigos e intenta averiguar el «truco»:

```
x = 4.34514513451
print(f"El número es {x :.2f}")
print(f"El número es {x :.4f}")
print(f"El número es {x :.0f}")
print(f"El número es {x :.5f}")
```

El número π comienza como 3.141592. Escribe el número π con varios decimales. Comprueba si al hacer esto Python redondea los números o simplemente escribe menos decimales.

6 Programas I

La suma de dos variables

El programa guarda dos números en variables y calcula su suma.

El doble y el triple

Se introduce un número en una variable y el programa calcula el doble y el triple del número.

Área de un rectángulo

Un programa que calcule el área de un rectángulo.

Área y perímetro de un círculo

Calcula el área y el perímetro de un círculo. La fórmula del área es πr^2 y la del perímetro es $2\pi r$.

7 Programas II

De kilómetros a metros

Un programa que pase de kilómetros a metros.

De pulgadas a centímetros

Un programa que pida una distancia en pulgadas y devuelva el resultado en centímetros. Debemos recordar que una pulgada son 2.54 cm.

Cociente y resto

Un programa donde se introduce el dividendo y el divisor y devuelva el cociente y el resto.

De grados centígrados a Farenheit

Un programa que pida la temperatura en grados centigrados y los transforme en grados Farenheit. Para convertir grados centigrados en Farenheit debemos multiplicar por 9 y dividir en 5. A este resultado se le debe sumar 32.

8 Cadenas I

Guardar cadenas en variables

Guarda tu nombre en una variable llamada `nombre` y tu apellido en una cadena llamada 'apellido'.

- Concatena las dos variables
- Escribe tu nombre 50 veces multiplicando la variable `nombre`.

Longitud de una cadena

Convierte el número 2^{50} a cadena y con la función `len()` calcula el número de dígitos que tiene.

† Poner en mayúsculas y minúscula una cadena

Prueba los siguientes códigos, para entender como funcionan algunos métodos que se pueden aplicar a cadenas:

```
s = "En un lugar de la Mancha de cuyo nombre no quiero ..."
s.upper()
```

```
s = "En un lugar de la Mancha de cuyo nombre no quiero ..."
s.lower()
```

```
s = "En un lugar de la Mancha de cuyo nombre no quiero ..."
s.title()
```

† Escribir carácteres que no aparecen en el teclado

Para realizar este ejercicio debemos consultar en Internet las tablas Unicode. Cada carácter tiene asociado un número. Podemos escribir, de una forma un poco extraña, cualquier carácter. Prueba los siguientes códigos:
(Si alguna vez no aparece la letra correspondiente prueba a escribir dicha cadena dentro de una función `print()`.)

```
"\u4567"
"Esto parece una letra de otro idioma: \u3245"
```

9 Cadenas II

Acceder a distintos caracteres de una cadena

Considera la cadena `Hola mundo`. Accede al primer y el último caracter utilizando índice positivos. Accede también utilizando índice negativos.

Slicing con cadenas

Dada la cadena `Física y Química`, almacenada en una variable 'dato' extrae la subcadena `Física` y la subcadena `Química`.

† Dar la vuelta a una cadena

Dada la variable `dato` del ejercicio anterior, prueba el siguiente código:

```
dato[ : : -1]
```

10 Operadores relacionales

Operadores relacionales y variables

Supongamos a lo largo del ejercicio que hemos declarado las siguientes variables:

```
a = 1
b = 5
c = 2
d = 1
```

Realiza, una a una, las siguientes comparaciones, intentado adivinar el resultado antes que Python:

```
a == b
b > a
a < b
a == d
b >= a
c <= b
d != a
d != b
```

Guardar resultados de comparaciones en variables

Adivinar el valor guardado en la variable `aprobado` después de ejecutar el código:

```
nota = 8
media = 7
aprobado = nota > media
```

Prioridad de los operadores aritméticos

Adivinar el resultado de la siguiente comparación y comprobar resultados en Python.

```
6 + 7 * 8 >= 5 / 3
```

Un resultado inesperado

Adivinar el resultado de la siguiente comparación. Comprueba con Python. Si el resultado no es el esperado intenta ver la razón.

```
3 * 0.1 == 0.3
```

No es buena idea utilizar el operador == cuando se trabaja con números reales.

El orden alfabético

Adivinar el resultado de las siguientes comparaciones y comprobar resultados en Python.

```
"hola" > "adios"
"benito" > "aratnxa"
"javier" >= "raquel"
"Hola" == "hola"
```

† El número asociado a cada letra o caracter

Todo caracter tiene asociado un número. En particular todas las letras tienen asociado un número. Para conocer el número asociado se utiliza la función `ord()`. Prueba los siguientes códigos:

```
ord(a)
ord(b)
ord(z)
ord(A)
ord(B)
ord(ñ)
```

† El caracter asociado a un número

A la inversa, muchos números enteros tienen asociado un caracter. Para obtener dicho caracter utilizamos la función `chr()`. Prueba los siguientes códigos:

```
chr(97)
chr(241)
chr(4456)
print(chr(4456))
```

Para entender completamente estos ejercicios se debe conocer a fondo la codificación de los caracteres, en particular es conveniente conocer la codificación ASCII y Unicode.

† El orden en las letras

En realidad Python compara números y no caracteres. Para saber si un caracter es menor que otro en realidad debemos fijarnos en su número asociado. Con lo visto en los ejercicios anteriores intenta predecir el resultado de la comparaciones:

```
'a' < 'A'
'z' < 'ñ'
'B' < 'a'
```

† Comparación de cadenas

Para comparar palabras Python compara la primera letra. Si la primera letra coincide, pasa a la segunda, etc. Comprueba los códigos:

```
"casa" < "cama"
"jamon" < "jabones"
```

11 Tipos de datos

Guardar y comprobar el tipo

Guarda tu nombre en una variable llamada `nombre`. Pregunta por el tipo de dato de la variable.

El tipo de una división

Guarda en una variable el resultado de la división de 4 entre 2. Comprueba que el de tipo `float`. El resultado de una división es siempre un número de tipo `float`.

El tipo del cociente de enteros

Calcula el cociente de la división de 4 entre 2. Comprueba su tipo.

El tipo de *True* y *False*

Calcula el tipo de `True`.

El tipo de las comparaciones

Guarda en una variable el resultado de la comparación `3 < 5`. Comprueba qué contiene dicha variable (con `print()`) y mira que tipo de dato contiene.

Python es un lenguaje de tipado dinámico

Guarda en un variable un tipo entero. Comprueba su tipo. Después guarda en la variable un tipo `float` y comprueba su tipo. Por último guarda una cadena y comprueba su tipo.

† El tipo complejo

Guarda en una variable llamada `complejo` el valor `1j`. Comprueba cual es su tipo de dato.
Calcula el cuadrado de dicha variable.

12 Entrada por teclado

Pedir el nombre y la edad

Un programa que pida el nombre y la edad del usuario y que dé una respuesta similar a:

```
Me llamo Joaquín y tengo 34 años.
```

Pedir dos números y realizar cuatro operaciones

El programa debe de pedir dos números (en principio valen números decimales) y el programa debe calcular su suma, su resta, su multiplicación y su división y devolverlas por pantalla.

Área y perímetro de un rectángulo

El programa debe de pedir la base y la altura de un rectángulo y calcular el área y el perímetro del rectángulo.

Usar variables dentro de la función *input()*

Podemos usar variables dentro de la función `input()` usando *f-strings*. Analiza el siguiente código, que pide dos números distintos al usuario:

```
num1 = int(input(f"Dime un número entero: )")
num2 = int(input(f"Dime un número distinto a {num1}. "))
```

13 Condicionales I

Fallo de indentación

Si después de los dos puntos no dejamos indentación, se produce un error. Prueba el siguiente código:

```
if True:
print("Esto es un error")
```

Fallo de indentación

Si dentro del bloque del `if` tenemos indentaciones distintas, también tenemos un error:

```
if True:
print("Hola")
print("Adiós")
```

Cambiar dos *if* por un *else...if*

El siguiente programa funciona perfectamente. Haz una versión similar utilizando `else`:

```
edad = int(input("Dime tu edad: "))

if edad >= 18:
print("Eres mayor de edad")
if edad < 18:
print("Eres menor de edad")
```

Error semántico

El siguiente programa no tiene errores de sintaxis, pero sin embargo tiene un error semántico (no hace lo que queremos que haga). Localiza el error:

```
edad = int(input("Dime tu edad: "))

if edad >= 18:
print("Eres mayor de edad")
if edad <= 18
print("Eres menor de edad")
```

Problema

Se pide un número al usuario. Si el número es mayor que cero el programa responde «El número es mayor que cero». En caso contrario debe responder «Este número no es mayor que cero».

Problema

Un programa pide el nombre de una persona. Si el nombre es «Javier» el programa responde «Eres un pro». Para cualquier otro nombre debe responder «Tu no eres un pro».

Problema

Un programa pide la nota. Si la nota es mayor o igual que 5 responde «Has aprobado». Si no ocurre esto responde «Has suspendido».

Problema

Se pide un número entero al usuario. Se calcula el resto de dicho número entre dos. Si el resto es cero, el programa responde «El número es par«. Si el resto no es cero el programa responde «El número es impar».

Problema

Se pide un número al usuario y se le dice si el número es múltiplo de tres o no lo es.

Problema

Haz un programa que pida dos números cuya multiplicación sea 12. Si el producto da 12 debe responder «Has acertado». En caso contrario debe responder 'Has fallado'. Realiza una versión del programa utilizando «else» y otra sin utilizarlo.

Problema

Se piden dos números enteros distintos. El programa debe responder si es mayor el primer número o es mayor el segundo número.

14 Condicionales II

Problema

Si piden dos números. El programa debe responder si el primero es el mayor, si es mayor el segundo o si los dos son iguales. Haz una versión sin utilizar `elif` y otra utilizándolo.

Problema

Se pide la nota de tres asignaturas. Si la media de las notas es mayor que 8, entonces el programa responde «Eres un pro». En otro caso debe responder «Tienes que trabajar más para ser un pro».

Problema

Un programa que pida la edad de una persona. Si la edad es mayor que 20 años, el programa debe responder «Eres muy viejo». Si no es mayor de 20 el programa no debe responder nada.

If anidado

Da un valor entero a la variable para que el siguiente programa imprima la palabra «Hola».

```
numero = ????
if numero > 3:
if numero < 10:
print("Hola")
```

¿Cuántos números enteros resuelven el problema anterior?

15 Condicionales III

Problema

Realiza un programa que diga si una persona es mayor de edad o no. Además debe comprobar primero si la persona tiene más de cero años.

Detectar error semántico

El siguiente programa tiene errores de tipo semántico. Localízalos y arregla el programa:

```
edad = int(input("Dime tu edad"))

if edad > 30:
print("Eres un adulto")
elif edad > 70:
print("Tercera edad")
elif edad > 15:
print("Eres joven")
else:
print("Eres un infante")
```

Problema

Un programa pide un número entero entre el 1 y el 3, ambos inclusive. Si se teclea 1 el programa responde «Primero». Si se teclea un 2 el programa responde «Segundo» y análogamente con el 3. Si no es ninguno de esos tres números, entonces debe responder «Número incorrecto».

Problema

Un programa pregunta por el dinero que tienes. Si la respuesta es mayor de 1 millón, el programa responde «Eres rico». Si tiene más de 30000 euros entonces debe responder «Clase media». En otro caso debe responder «Clase baja».

16 Operadores lógicos

Tabla de verdad del operador *and*

Calcula la tabla de verdad del operador `and`.

Traducir al lenguaje Python

Traducir la siguiente frase a Python:

```
4 es menor que 5 y 8 es menor que 13
```

Traducir

Sea `p` la variable que contiene la frase «El sol es una estrella», y `q` la variable que contiene «La tierra es un planeta». Traduce al lenguaje español las siguientes expresiones:

```
p and q
p or q
not q
p and not q
```

Traducir

Sea `x` una variable. Traduce a Python las frases:

> `x` es menor que 5 y `x` es mayor que 0
>
> `x` es mayor que 20 y ademas es menor que 50
>
> `x` es igual a 20 o `x` es menor que 100

Comprueba dichas traducciones dando distintos valores a la variable `x`.

Traducir

Una persona es adolescente si tiene entre 12 y 18 años. Utiliza una variable `edad` y crea una expresión que sea verdadera únicamente para los adolescentes.

Operación combinada

Realiza paso a paso la operación lógica combinada:

```
True and False or not True
```

† Otra forma de realizar los operadores booleanos

Aunque la forma normal de realizar los operadores booleanos es utilizar las palabras clave `and` y `or` también se pueden realizar con otros operadores. Para hacer la operación `and` podemos usar `&` y para el `or` el caracter `|`.
Calcula todas las tablas de verdad de estas operaciones. Como ejemplo tenemos el código:

```
True & False   # Equivale a True and False
True | False   # Equivale a True or False
```

El operador 'xor'

La operación `xor` es otra operación entre valores lógicos. Se suele denominar **o exclusiva** pues cuando los dos valores son ciertos entonces devuelve `False`. El operador para realizarla es el circunfleno `^`.

```
True ^ False  # Se lee: True xor False
```

Calcula la tabla de verdad de este operador.

† Un atajo para los condicionales

Dadas tres variables es muy común, tanto en matemáticas como en Python hacer el siguiente tipo de comparaciones:

```
x < y and y < z
```

Estamos comprobando si la variable y está entre las otras dos variables. Esto se suele escribir en matemáticas y también en Python con la siguiente notación, más compacta:

```
x < y < z
```

Además de con el operador < existe otros atajos con el resto de operadores relacionales.
Encontrar todos los valores enteros de la variable y que hacen que el siguiente programa tenga `True` como resultado:

```
x = 5
y = ???
z = 10

x <= y <= z
```

17 Condicionales IV

Problema

Para ser directivo de una empresa debes tener más de 25 años y además más de 30000 euros. Haz un programa que pida la edad y el dinero y que responda si puedes ser directivo de esa empresa.

Problema

Para ir a un viaje a Canarias tienes que tener más de 65 años o tener más de 1000 euros. El programa pregunta la edad y el dinero y debe responder si puedes ir o no a Canarias.

18 Listas I

Acceder a elementos de una lista

Dada la lista `a = [4, 6, 7, 8]` accede a distintos elementos utilizando tanto índices positivos como negativos.

Acceder a elementos de una lista

Dada la lista `a = [4, 6, 7, 8]` adivina el resultado de:

```
a[0] + a[-1]
a[2] ** 2
a[0] * a[1] * a[2] * a[3]
```

†[⇑] Los índices tienen que ser enteros

Si el índice es un número decimal, entonces tenemos un error:

```
a = [1, 2, 3, 4]
a[1.0]
```

Una de las siguientes instrucciones dará un error. Explica la razón.

```
a[6 / 3]
a[6 // 3]
```

Crear listas con el mismo elemento

Crea una lista con 100 elementos y todos deben ser iguales a 0.

La lista nula

En cierto sentido, la lista nula se comporta como el número cero. Responde a las siguientes cuestiones para comprobarlo.

- Calcula la longitud de la lista nula.
- Concatena la lista nula consigo misma y observa el resultado.
- Multiplica la lista nula por un número y observa el resultado.
- Suma la lista nula a otra lista y observa el resultado.

Multiplicación y concatenado

Intenta predecir el resultado del código:

```
a = [1, 2]
b = ['a', 'b']
a * 3 + b * 2
```

† Convertir cadenas en listas

La función `list()` puede convertir una cadena en una lista:

```
cadena = 'Matemáticas'
l = list(cadena)
print(l)
```

Elementos en listas

¿Qué resultado obtenemos del siguiente código?

```
jinetes = [?guerra?, ?hambre?, ?peste?, ?muerte?]
?peste? in jinetes
?libertinaje? in jinetes
```

† Listas anidadas

Una lista puede conter cualquier tipo de dato. En particular puede contener a otras listas. Cuando esto sucede se dice que las listas están **anidadas**.

```
a = [ [1, 2], ['a', 'b']]
a[0]  # Accede al primer elemento que el una lista
a[0][1]  # Accede a la posición 1 del primer elemento
```

† Más listas anidadas

Considera la siguiente lista x formada por «tres filas»:

```
a = [1, 2, 3]
b = [4, 5, 6]
c = [7, 8, 9]
x = [a, b, c]
```

- Utilizando *slicing* accede al elemento 3 y al elemento 8.
- Adivina el resultado del siguiente código:

```
x[2][0] + a[-1] - b[2]
```