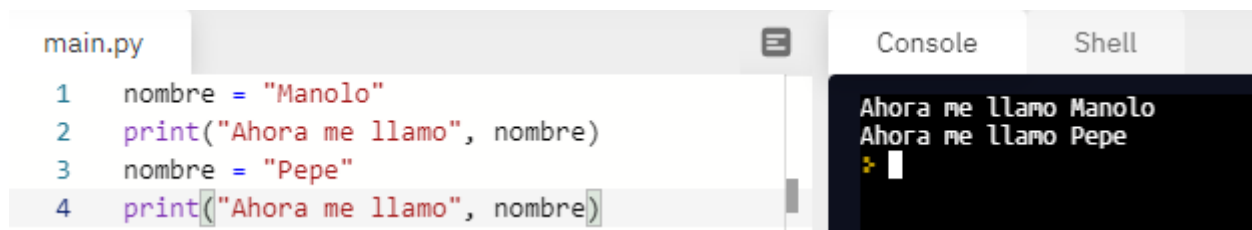


Variables

Variables, tipos y convenciones

Asignación de valores a las variables

Asignar un valor a una variable es hacer que la variable guarde este valor. Ese valor se ASIGNA usando el símbolo igual (=)



```
main.py
1 nombre = "Manolo"
2 print("Ahora me llamo", nombre)
3 nombre = "Pepe"
4 print("Ahora me llamo", nombre)
```

Console Shell

```
Ahora me llamo Manolo
Ahora me llamo Pepe
>
```

Aquí a la variable nombre primero le asigno el valor manolo, hasta que en la línea 3 le asigno otro valor. Las variables toman el último valor que se les asigne.

Otro ejemplo de asignación

En este caso, cojo el valor que tenía la variable nombre, y le añado otra palabra. Esto también es muy habitual. En este caso el + no sirve para sumar, puesto que se trata de texto.



```
main.py
1 nombre = "Manolo"
2 print("Ahora me llamo", nombre)
3 nombre = nombre + " Pepe"
4 print("Ahora me llamo", nombre)
5
```

Console Shell

```
Ahora me llamo Manolo
Ahora me llamo Manolo Pepe
>
```

Tipos de variables

Hablemos un poco en detalle de los tipos, y así podré dejar de poner "palabras" entre comillas...

Uno de los tipos de variables que más usaremos son los enteros (integer en inglés, abreviado como int con frecuencia). Ya sabéis número sin decimales, positivos y negativos.

Si necesitamos números decimales, usaremos "punto flotante" (float).

Tenemos las cadenas de caracteres (en inglés string), conjuntos de símbolos que incluyen las letras mayúsculas y minúsculas, los signos de puntuación, los caracteres especiales, etc.

También se puede tener una variable en la que se vaya a guardar sólo un carácter

(char)

Las hay que almacenan nada más que un bit, un uno o un cero (boolean) y algunos más...

Aquí podéis ver algunos ejemplos. Lo que está después de # son comentarios que no hacen nada en el programa. Solo explican qué tipo de dato hay en cada variable.

```

main.py  saving...
1  edad = 3 #int
2  pi = 3.14 #float
3  nombre = 'Daniel' #cadena o string
4  inicial = 'D' #carácter
5  casado = True # Booleano

```

Según cómo escribimos las cosas, Python interpreta si esa variable contiene un número, una letra, un decimal, una palabra, etc. En nuestro ejemplo, la variable nombre era de tipo string.

¿Cómo podemos llamar a nuestras variables?

Ya has visto que llamamos a la variable como mejor nos pareció, pero hay ciertas reglas.

No podemos usar palabras “reservadas” del propio lenguaje de programación (print, input... no son nombres de variables permitidos)

Deben consistir en una sola palabra (edad del usuario no es un nombre válido)

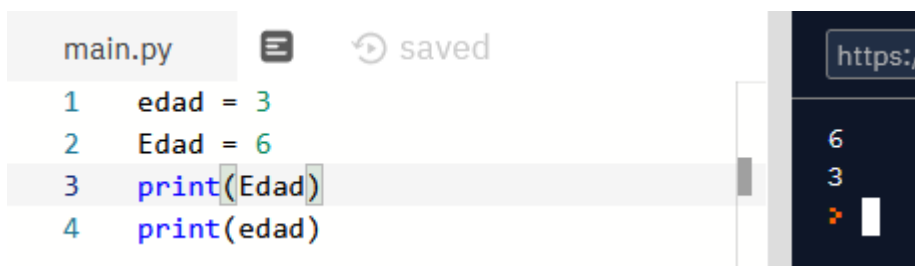
Sólo pueden contener letras, números y el guion bajo

No pueden empezar con un número

Mayúsculas y minúsculas

Aunque se pueden usar mayúsculas, y Python es muy cuidadoso con eso, entendiendo que Nombre, NOMBRE y nombre son variables DISTINTAS, lo mejor que podemos hacer es no liarnos siguiendo ciertas recomendaciones.

Fijaos en el ejemplo:



```

main.py  saved
1  edad = 3
2  Edad = 6
3  print(Edad)
4  print(edad)

```

The output of the script is shown in a terminal window on the right:

```

https://
6
3

```

También es típico empezar con minúsculas, como hicimos en nombre. Y usar alguno de estos dos estilos para separar palabras:

```

main.py  saved
1  numeroCarnet = '43156667X'
2  nombre_Hijo = 'Calamardo'

```

Consejos

Los acentos suelen dar problemas por lo que mejor evitarlo.

Para la ñ hay quien para no escribir año usa anio , o bien, anyo .

En programas elaborados es muy conveniente elegir nombres de variables que nos den una pista de qué están representando

Ejemplo nombre y edad

Vamos a hacer un programa que pida al usuario su nombre y año de nacimiento para decirle al final:

Hola, Paco, tienes 45 años.

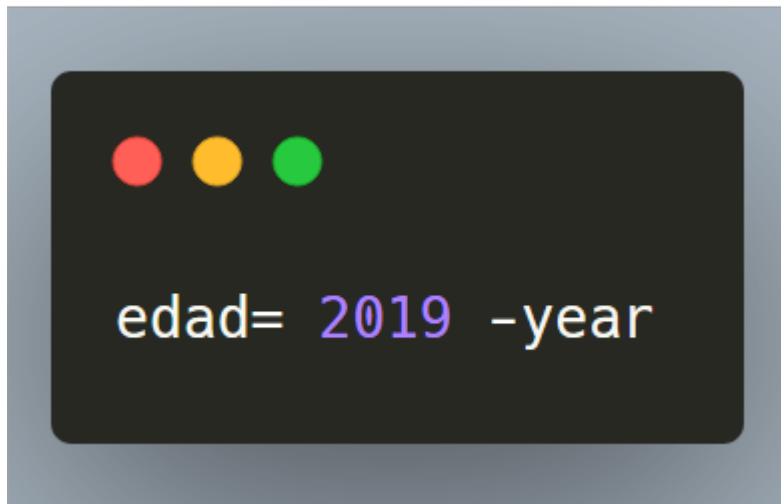
Para pedirle el nombre y el año de nacimiento usamos la misma estructura que antes. Una línea con un PRINT para que sepan lo que queremos y otra línea con un INPUT para recoger lo que nos han dicho.

Todas las líneas que pongo aquí las tienes que escribir unas debajo de las otras en el mismo ejercicio. Va todo junto.

```
print('¿Cómo te llamas')
nombre = input()
print('¿En qué año naciste?')
year = input()
```

Estupendo. Ahora cómo calculamos la edad. Bueno, pues sabemos que la edad será el año actual menos el año de nacimiento.

Pero fíjate, esa resta nos da un valor, que es justo lo que queremos, pero que...




Cálculos con variables

El siguiente paso será calcular nuestra edad.

Con esto creamos la variable edad y le ASIGNAMOS el valor de la resta entre el año 2019 y el valor que tenga la variable year , que nos acaba de dar el usuario, y QUE SERÁ DISTINTO EN CADA EJECUCIÓN.

Pintar información por pantalla




```
print ( 'Hola' ,nombre, 'tienes' ,edad, 'años' )
```

Escribe y ejecuta el código anterior junto y adjunta una captura de pantalla. Te dará error, pero es normal. Luego lo arreglaremos

Arreglar errores

Al ejecutar el programa veremos que nos aparece en rojo un error. LOS ERRORES SON NORMALES. Leamos el mensaje de error.



```
Traceback (most recent call last):  
File "main.py", line 8, in <module>  
edad=2019-year  
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

La línea 8 es donde yo tengo escrita la operación `edad= 2019 -year`. En la última línea me dice que el operador "menos" no puede trabajar con un tipo entero y un tipo string a la vez.

El asunto es que TODAS LAS ENTRADAS POR TECLADO se toman como CADENAS DE CARACTERES.

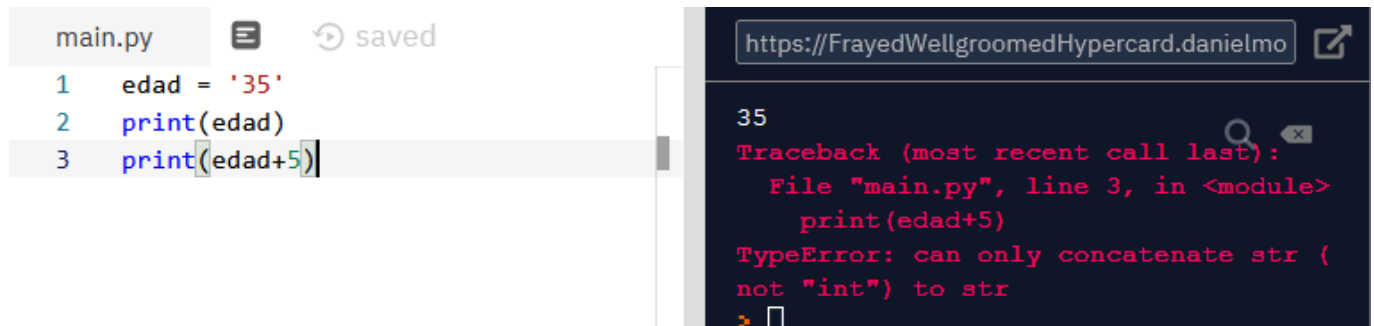
Así que el año que introdujo el usuario, para Python son LETRAS (caracteres), y no sabe cómo restar letras y números.

Convertir letras en números

Las variables pueden ser de diferentes tipos, y podemos transformar una variable de un tipo en otro. En este caso es necesario, puesto que para hacer operaciones matemáticas hay que convertir las letras en números.

¿Hay una manera en Python de tomarse 45 como si fuera un número o como si fuera una cadena de caracteres? Sí.

Hay que usar la función `int()`. Convierte, si se puede, una cadena de caracteres en un número entero. Veamos el siguiente ejemplo. Edad contiene un string, es decir, las "letras" 3 y 5. Por eso va entre comillas. Cuando queremos sumar una palabra y un número no podemos (no es posible). Los dos objetos a sumar deben ser números o floats (decimales), por eso falla.



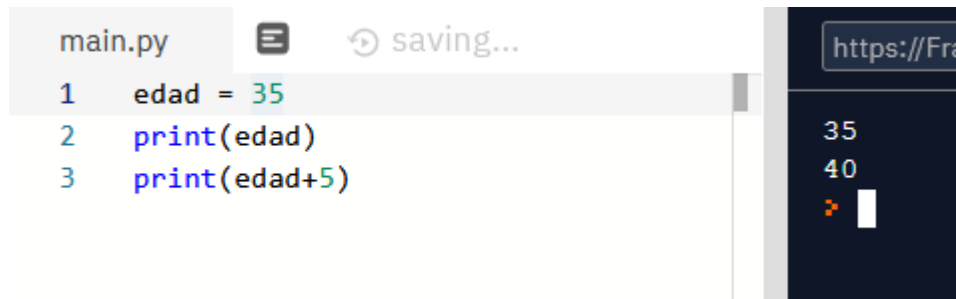
The screenshot shows a code editor with a file named `main.py` containing three lines of Python code:

```
1 edad = '35'  
2 print(edad)  
3 print(edad+5)
```

The code is saved. To the right, a terminal window shows the output of the first line, `35`, followed by a `Traceback` error message:

```
Traceback (most recent call last):  
  File "main.py", line 3, in <module>  
    print(edad+5)  
TypeError: can only concatenate str (not "int") to str
```

En este otro caso, veréis que la variable `edad` no lleva comillas. Ahora está guardando 35 como número, no como palabra. Por ello, cuando quiero sumar la variable `edad` y el número 5, al ser ambos números, funciona.



The screenshot shows the same code editor with the code from the previous example, but now the variable `edad` is assigned the integer value 35:

```
1 edad = 35  
2 print(edad)  
3 print(edad+5)
```

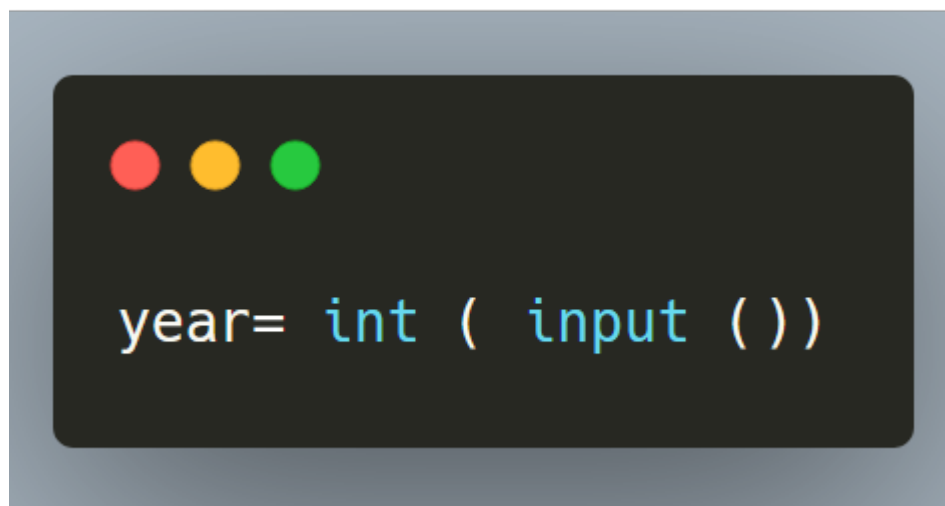
The code is saving. The terminal window shows the output of both lines:

```
35  
40
```

Convertir tipos de variables

Cambiar de un tipo a otro se llama **casting**. Si queremos hacer una operación y necesitamos convertir una variable de un tipo a otro, podemos utilizar unas funciones que los transforman.

En nuestro caso, podríamos hacer:

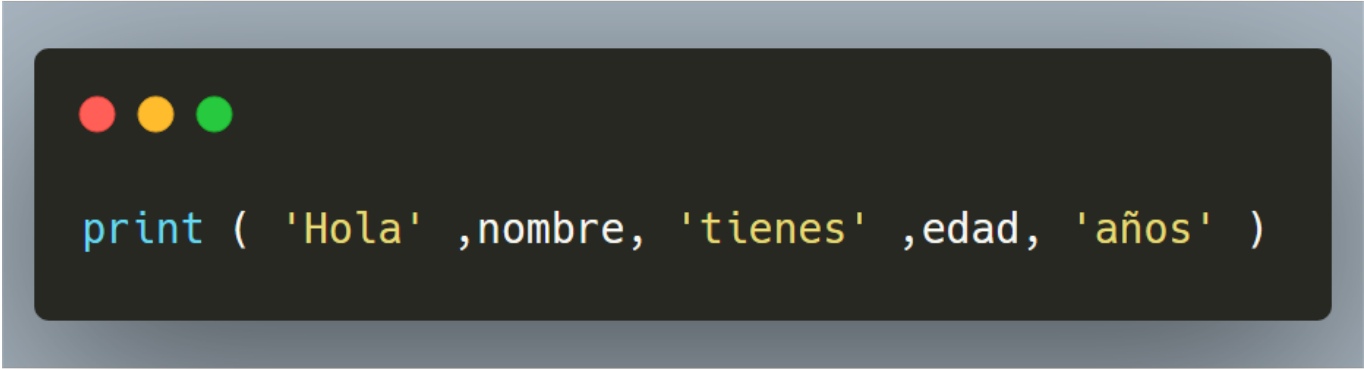


The screenshot shows a code editor with a single line of Python code:

```
year= int ( input ( ))
```

Lo que significa: toma lo que pongan en el teclado, conviértelo en un número entero y almacenarlo como tal en la variable `year`.

Finalmente, si queremos una salida por pantalla, podríamos poner:



```
print ( 'Hola' ,nombre, 'tienes' ,edad, 'años' )
```

Por supuesto, si haces algo como `int(Paco)` aparecerá un error.

Actividad 2. Escribe y ejecuta el código anterior junto y adjunta una captura de pantalla. Ya debería funcionar y dar el resultado deseado

Comentarios

Hagámonos una pregunta, ¿después de unas semanas te acordarás qué significaba esta línea?

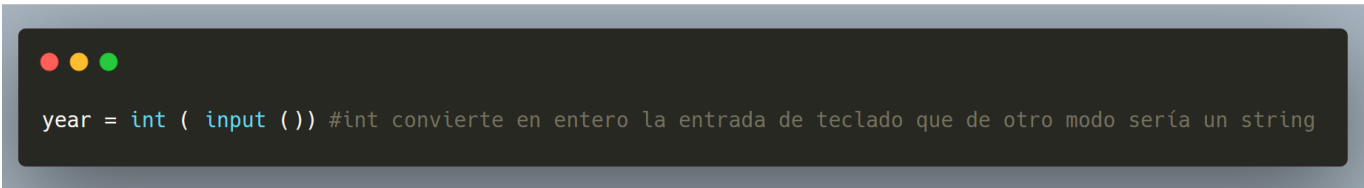


```
year= int ( input ())
```

Sería estupendo poder incluir algo en los programas que nos explicara qué están haciendo, pero si escribimos texto en castellano, Python nos dará un error porque se pensará que son instrucciones de Python.

El símbolo `#` nos permite añadir comentarios al programa y que Python sepa que debe ignorar lo que va después de la almohadilla.

Por ejemplo, en nuestro programa podríamos.



```
year = int ( input ()) #int convierte en entero la entrada de teclado que de otro modo sería un string
```

Al leer los comentarios, entendemos el programa si no es nuestro, o recordamos qué habíamos hecho si somos los autores.

Con esto en mente el programa de antes quedaría así:

```
'''Pregunta nombre y año de nacimiento y calcula la edad'''
print ( '¿Cómo te llamas?' ) #saca por pantalla esta pregunta
Nombre = input () #lo que se escriba en el teclado se mete en la variable nombre
print ( '¿En qué año naciste?' ) #saca por pantalla esta pregunta
year = int ( input () ) #int convierte en entero la entrada de teclado que de otro modo sería un string
edad = 2019 - year #calculamos la edad restando del año actual
print ( 'Hola' , nombre, 'tienes' , edad, 'años' ) #sacamos por pantalla los resultados
```

Quizá te suene muy liso, sobre todo porque estamos explicando cosas que ya son bien conocidas. Igual podíamos dejarlo en:

```
'''Pregunta nombre y año de nacimiento y calcula la edad'''
print ( '¿Cómo te llamas?' )
nombre = input () #nombre usuario
print ( '¿En qué año naciste?' )
year = int ( input () ) #int convierte en entero la entrada de teclado que de otro modo sería un string
edad = 2019 - year #edad usuario (en el 2019)
print ( 'Hola' , nombre, 'tienes' , edad, 'años' )
```

Más sencillo y con información suficiente para entenderlo.

Has podido ver que empezamos el programa explicando qué hace. Aquí también se puede añadir información del autor, fecha, contacto y otros detalles que se consideren relevantes.

Has visto también que cuando vamos a poner un comentario que ocupa más de una línea usamos tres comillas sencillas.

Actividad 3. Modifica el código anterior para que calcule el año en que te vas a jubilar (supuestamente) a partir de tu edad. Piensa que hoy en día la edad de jubilación es 65. En mi caso, yo le escribiré 35 años con el teclado y el me dirá que me quedan 30 años para la jubilación. Adjunta una captura

Operaciones matemáticas

Ya hemos visto que podemos restar una variable numérica de otro número, también las podemos sumar, multiplicar o dividir, pruébalo.

Hay otras operaciones muy comunes en programación que merecen mencionarse.

Módulo , resto, residuo...

Es el resto de una división. Si dividimos 7 entre tres, dará como cociente 2 y como resto 1. Escribe en la consola `7%3` y verás que da como resultado 1

División entera

Nos da el cociente de la división sin obtener decimales. Si escribes en la consola `7 // 3` obtendrás 2 En cambio si escribes `7 / 3` obtendrás 2.333333333

Potencias

Si escribes en la consola $2^{**}3$ obtendrás 8 (dos elevado a tres). El asterisco sólo te dará la multiplicación $2*3$ será 6

Imaginemos que nuestro personaje sube de nivel y necesita una cantidad de experiencia para subir.

Nivel 1 -> $10*2^1 = 20$

Nivel 2 -> $10*2^2 = 40$

Nivel 3 -> $10*2^3 = 80$

Nivel 4 -> $10*2^4 = 160$

Orden de operaciones

Para las operaciones hay un orden de prevalencia, como en las matemáticas comunes, por lo cual debes tener cuidado y usar paréntesis, así como asegurarte de que lo que escribes funciona como deseas.

Las operaciones con números te son conocidas... pero también pueden hacerse operaciones con cadenas de caracteres.

Aquí tienes un resumen con los símbolos de las principales operaciones matemáticas:

Operador	Tipo	Operación que realizan
+	Matemático	Sumas datos numéricos
-	Matemático	Restar datos numéricos
*	Matemático	Multiplicar datos numéricos
/	Matemático	División de 2 valores numéricos
//	Matemático	División entera, toma la parte entera del resultado
**	Matemático	Potenciación, para obtener un número elevado a un exponente
%	Matemático	Resto, módulo o residuo, se obtiene el resto de la división

Actividad 4. Escribe un programa en que dados dos números (el día de mes de tu cumpleaños y el número de personas que viven en tu casa), muestre:

El cociente entero (sin decimales)

El resto

La división exacta

La potencia del primer número elevado al segundo

En mi caso estos números son 13 (13 de septiembre) y 3 (mi mujer y mi hija) y lo que saldría por pantalla es:

4

1

3,25

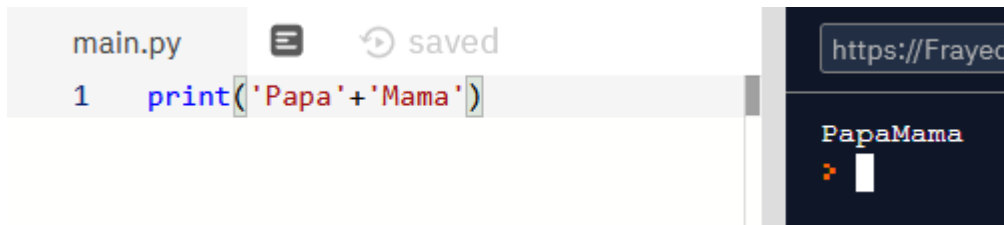
13.013

Adjunta una captura de la pantalla

Operaciones con cadenas de caracteres

Concatenación

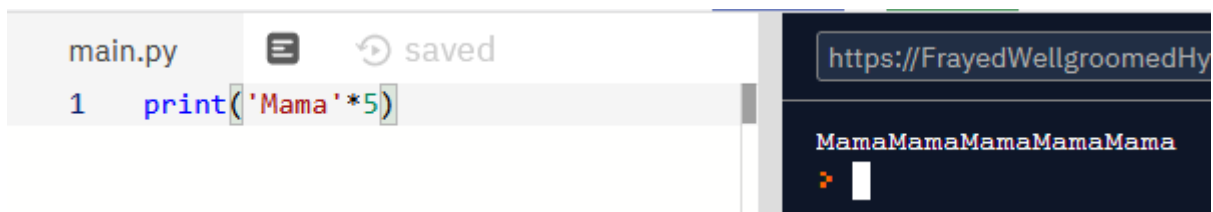
Por ejemplo, ¿qué pasará si escribes 'Mamá' + 'Papá'? Prueba... Efectivamente, concatena (junta) las dos cadenas, haciendo una cadena mayor.



The screenshot shows a code editor with a file named 'main.py'. The code contains a single line: `1 print('Papa'+'Mama')`. To the right of the code editor is a terminal window showing the output of the program: `PapaMama`.

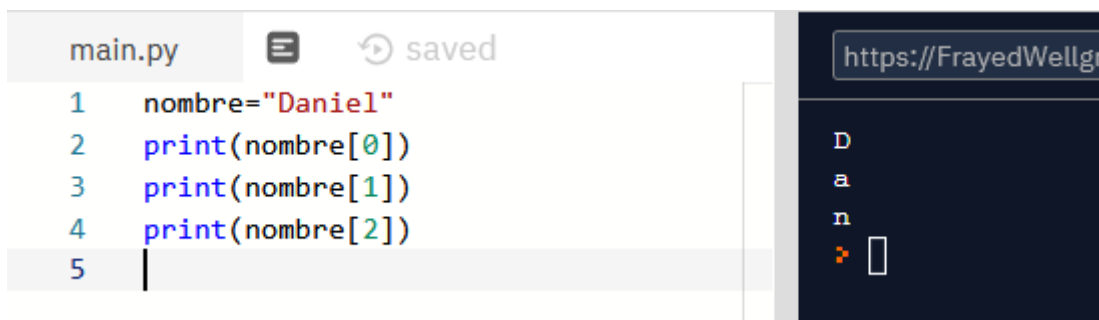
Repetición

Intenta hacer 'Mamá'*5 a ver qué pasa...



The screenshot shows a code editor with a file named 'main.py'. The code contains a single line: `1 print('Mama'*5)`. To the right of the code editor is a terminal window showing the output of the program: `MamaMamaMamaMamaMama`.

También podemos obtener partes de un string. Para ello utilizaremos []. Si dentro colocamos un número, nos dará la letra que ocupa esa posición. En informática, comenzamos a contar por el 0. Por lo tanto, la D ocupará la posición 0, y la a la posición 1, etc.



The screenshot shows a code editor with a file named 'main.py'. The code contains four lines: `1 nombre="Daniel"`, `2 print(nombre[0])`, `3 print(nombre[1])`, and `4 print(nombre[2])`. To the right of the code editor is a terminal window showing the output of the program: `D`, `a`, and `n` on separate lines.

Actividad 5. Crea un programa que te pida tu nombre y tus dos apellidos uno a uno, los guarde en 3 variables y te pida por pantalla las iniciales. En mi caso, saldría por pantalla DMR. Necesitarás utilizar las operaciones anteriores y lo estudiado anteriormente