CREAR CON PYTHON

MATERIALES DE INICIO A LA PROGRAMACION CON CODIGO

fundación esplai
ciudadanía comprometida

red conecta
conecta joven

Con el apoyo de
Microsoft

Financiado por
POR SOLIDARIDAD
OTROS FINES DE INTERÉS SOCIAL

fundaciónesplai ciudadanía comprometida | ConectaCode

1.Conoce Python

Qué, porqué, cómo

2,Instalar Python

Descargar e instalar Python

3.Python y IDLE

Trabajar con IDLE

4.Programas de robots

Algoritmos en acción

5.Python y su tortuga

Robots que dibujan

6.Variables

Almacenar datos

7.Utilizando números

Python y las matemáticas

8.Cadenas y entradas

Palabras y teclas del teclado

9.Bucles

Hacer algo una y otra vez

10.Mostrar por pantalla

Como imprimir por pantalla

11.Listas

Tener los datos ordenados

12.Cierto o falso

Tomando decisiones

13.Bifurcaciones

Utilizando If y If-Else

14.Bucles tipo While

Bucles y condiciones

15.Funciones

Reutilizar código una y otra vez

1. CONOCE PYTHON

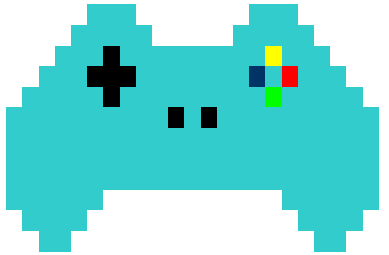
¿Qué son los programas de ordenador? ¿Por qué son importantes? ¿Por qué aprender a crearlos? Quizás te sorprenda saber que los programas informáticos hacen que el mundo sea como es hoy día.

Lo que aprenderás

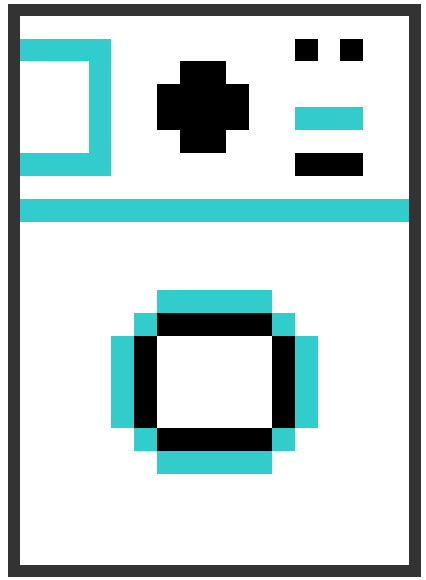
- Los ordenadores forman parte de tu día a día
- Los ordenadores ejecutan programas que les dicen qué hacer
- Los programas se escriben en un lenguaje especifico
- Python es un lenguaje muy adecuado para iniciarse
- ¡Programar es muy divertido!

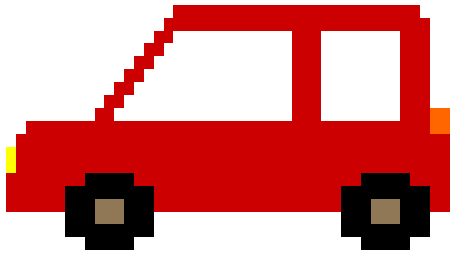
QUE ES UN PROGRAMA DE ORDENADOR

Los ordenadores se encuentran en todo nuestro entorno pero a menudo no sabemos que están allí. Ya conoces los ordenadores de escritorio y portátiles, pero también hay ordenadores dentro de nuestros teléfonos y tabletas, en consolas de juegos e incluso en televisores, lavadoras y automóviles.



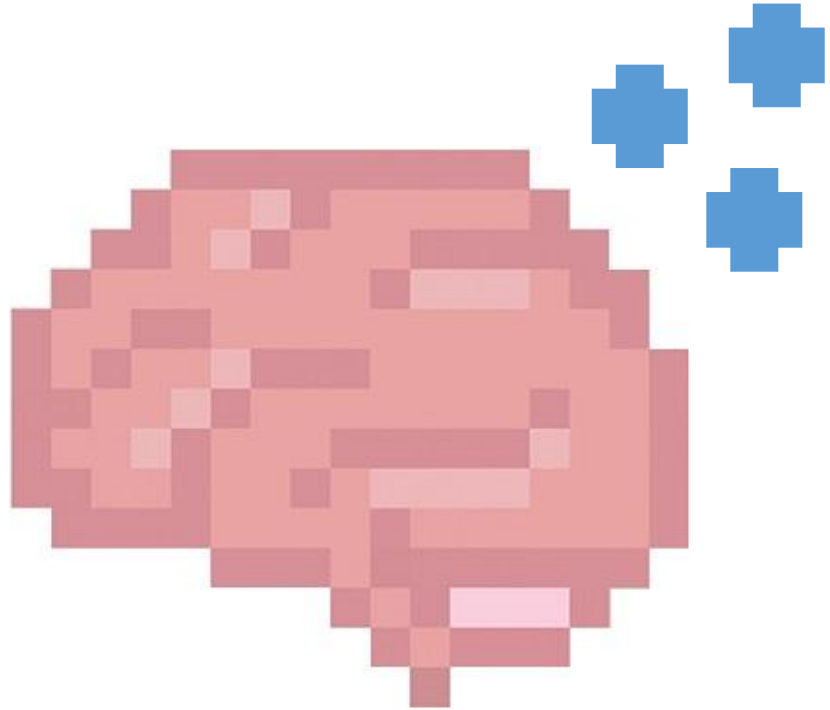
Los ordenadores llevan a cabo listas de instrucciones llamadas programas. Los videojuegos, los navegadores web y los procesadores de texto, son todos programas de ordenador. Tal vez, te sorprenda que también los ascensores, las lavadoras, y los frenos de un coche están controlados por programas de ordenador.





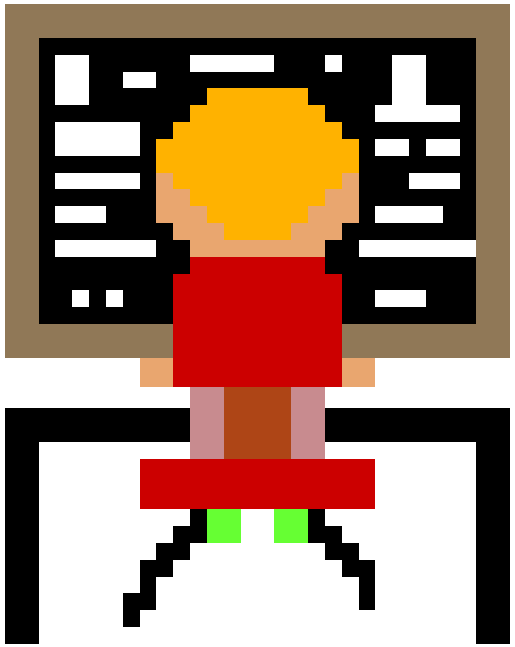
POR QUE APRENDER A PROGRAMAR

Si aprendes a crear programas, puedes tomar el control de los ordenadores que te rodean. Conviértete en un programador, y podrás hacer tus propios juegos, controlar tus propios robots y escribir tus propias aplicaciones para dispositivos móviles. Lo más importante: la programación es un reto divertido, y es un buen ejercicio para ejercitar la mente.



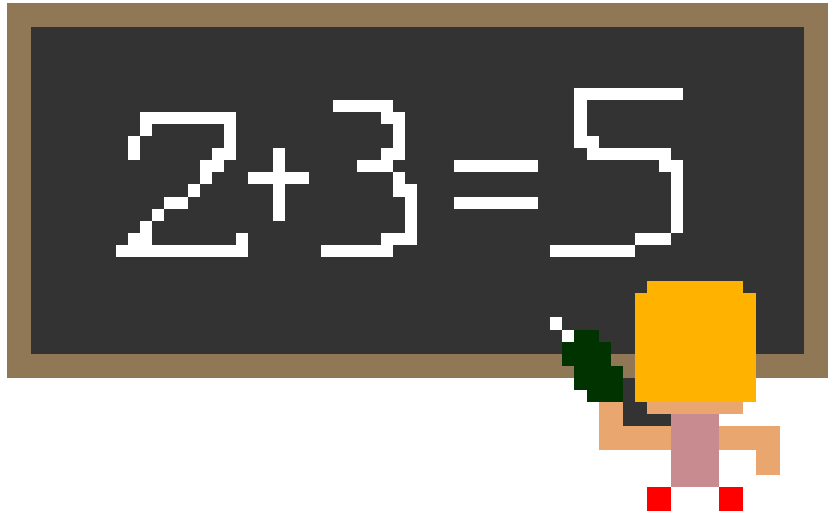
COMO COMENZAR A PROGRAMAR

No puedes decirle a un ordenador simplemente: "juega al ajedrez". Los ordenadores no saben nada de nada. En su lugar, el programador debe decirle al ordenador todo lo que necesita saber sobre el ajedrez en trozos pequeños y simples de información. Los ordenadores no entienden el lenguaje normal, se les debe decir mediante instrucciones especiales conocidas como **lenguaje de programación**.



EL LENGUAJE DE PROGRAMACION

Las instrucciones que le dicen a un ordenador cómo realizar una cierta tarea, **necesitan ser codificadas en un lenguaje** que sea bastante simple para que los ordenadores interpreten y además puedan ser entendidas por los seres humanos (para que puedan hacer el programa). Entonces "ejecutamos" nuestro programa para que el ordenador realice la tarea.

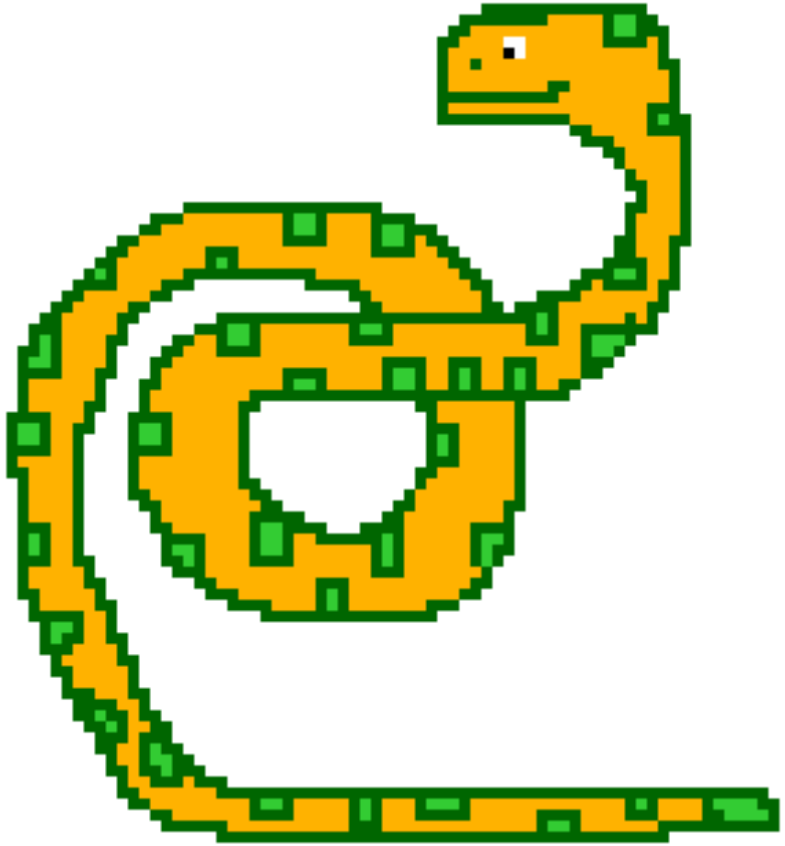


Un ordenador podría hacer un problema matemático como éste, pero primero tendría que saber cómo hacerlo.

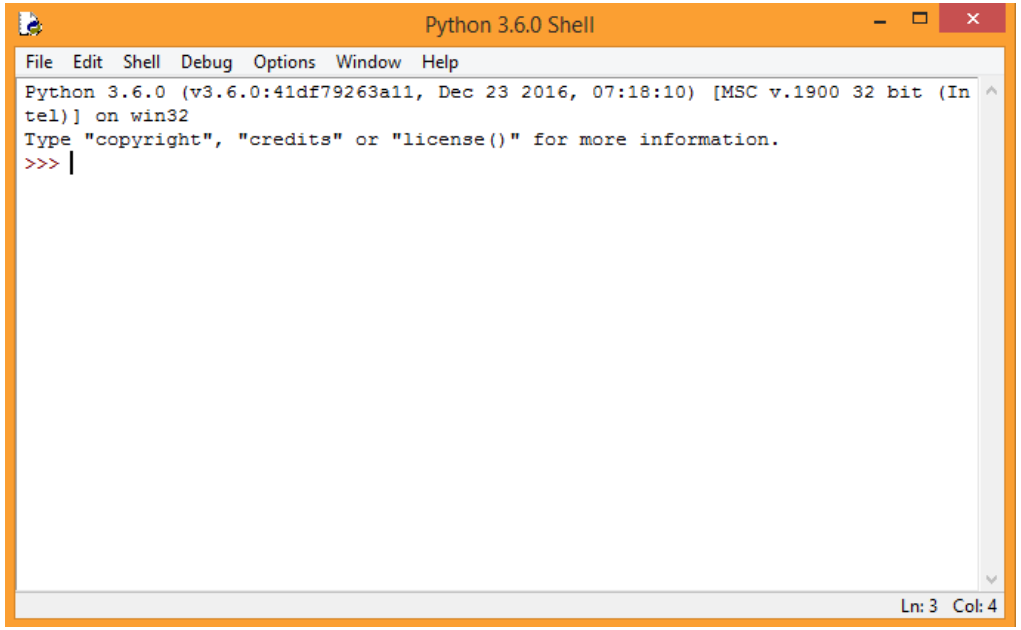
Hay muchos **lenguajes de programación** diferentes, diseñados para permitir que los ordenadores realicen una amplia gama de tareas. Algunos tienen nombres que suenan extraños, tales como Java, C++ o Ruby.

PYTHON NO ES UNA SERPIENTE

Python es un lenguaje de programación. Aunque es fácil de aprender, se puede utilizar para escribir todo tipo de programas. Python se utiliza no sólo en las escuelas, sino también en empresas y universidades de todo el mundo.



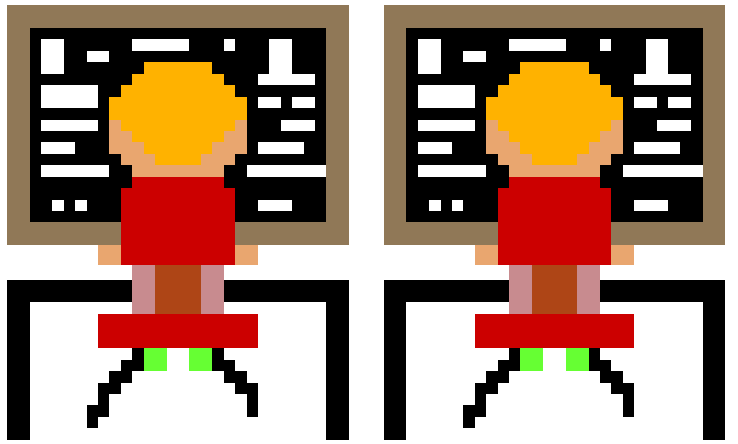
Python está diseñado para que sus programas sean fáciles de leer y entender. Es un gran primer lenguaje para aprender, y a medida que vayas mejorando como programador, serás capaz de acometer proyectos cada vez más complejos. Aquí vas a conocer las bases de Python, pero los conceptos que aprenderás son comunes a cualquier lenguaje de programación.



Ésta es la ventana donde vamos a escribir el código de Python (Python 3.6.0 Shell)

APRENDER CON ESTE TUTORIAL

¡Comienza ya con la programación en Python! Cada nuevo concepto se explica usando ejemplos prácticos para ayudarte a entender y recordar cómo funciona esa parte de Python.
Te animamos a que pruebes el código de los ejemplos en tu ordenador, pero antes tendrás que instalar Python en tu PC.



2. INSTALAR PYTHON

Para seguir correctamente este tutorial, tendrás que instalar Python 3 en tu ordenador. Esto te permitirá ejecutar el código de los ejemplos y problemas que se plantean.

Recuerda que a programar, se aprende programando. ☺

Lo que aprenderás

- Qué es IDLE
- Cómo instalar Python 3 y IDLE en el ordenador
- Python 2 no nos sirve

TRABAJAR CON IDLE DE PYTHON 3

Vais a utilizar un programa llamado IDLE, que facilita el uso de Python. IDLE se instala junto con Python. Así que vas a comenzar por instalar Python e iniciar IDLE con las siguientes instrucciones para tu ordenador.

Quizás ya tienes Python instalado en el equipo, pero comprueba que es Python 3, y no Python 2. Cuando inicies IDLE, el número que verás después de "Python" en la primera línea debe ser un 3, no un 2. En caso de duda, sigue las instrucciones para instalar Python 3.

INSTALAR PYTHON

El entorno de programación de Python está disponible en para varios sistemas operativos que puedes descargar desde la web oficial http://www.python.org/downloads

Nosotros nos centraremos en la versión de Windows. En el momento de elaborar este tutorial la versión más actualizada en la 3.6.1.

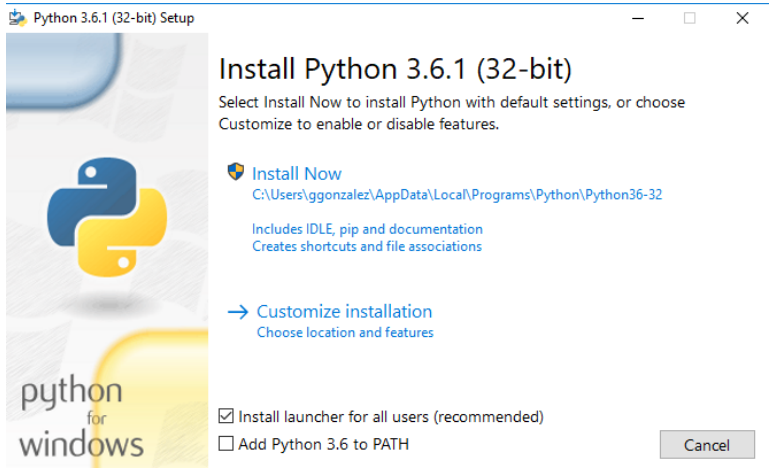
¿Cómo descargarlo e instalarlo en tu PC?

1. Acceder a la web de Python:
www.python.org/downloads/

2. Descargar el instalador de Python:
Download Python 3.6.1.

3. Instalar Python:
Una vez hayas descargado el paquete, debes hacer doble clic sobre el fichero, pulsar en “Install Now” y seguir los pasos que indica el instalador.

En el caso de que tu PC tenga MacOS o Linux como sistema operativo deberás descargar el instalador correspondiente en la web.

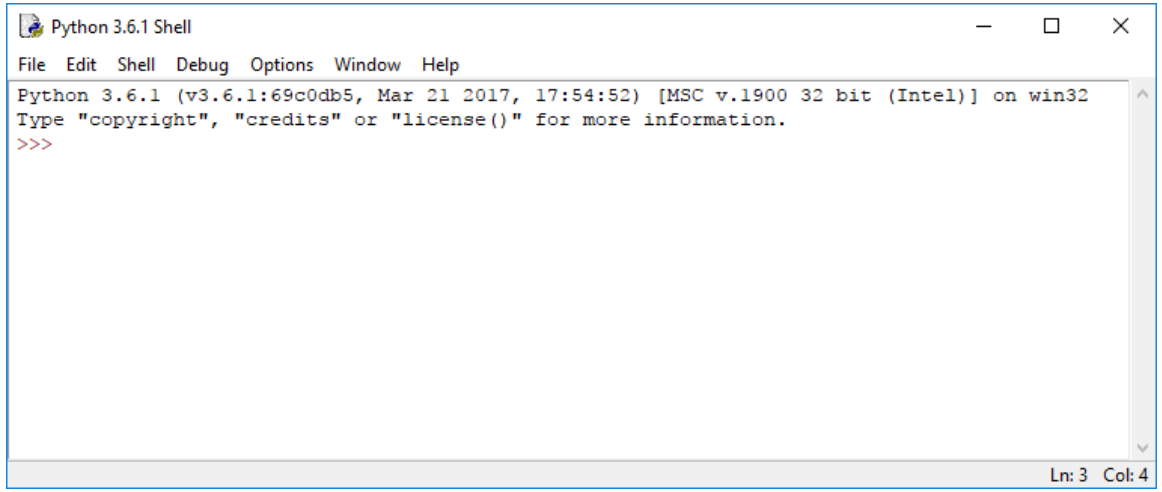


4. Ejecutar IDLE

Una vez se haya instalado Python, vas a ejecutar el programa llamado IDLE, que no es ni más ni menos que el entorno donde vamos a escribir el código de Python. Si no se ha creado un icono en el escritorio, utiliza le herramienta de buscar que incluye Windows.

5. La ventana de Python

Una vez ejecutes IDLE, se va abrir la ventana donde se procederá a programar en lenguaje Python. A esta ventana se le llama Shell.
En la imagen puedes observar que en la última línea aparecen los símbolos >>> que es donde empezaremos a programar.



3. IDLE Y PYTHON

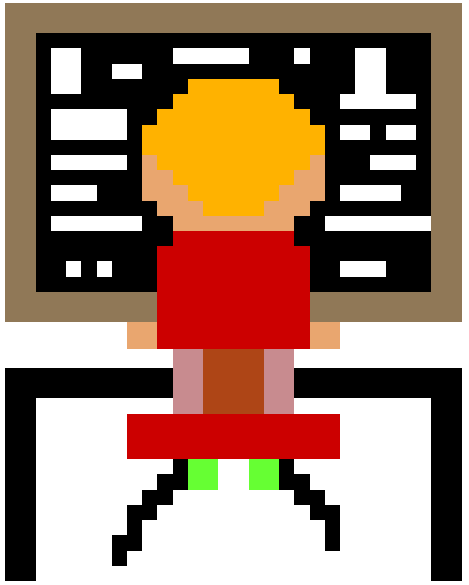
Ahora que tienes instalado Python 3 y su entorno de trabajo IDLE en el ordenador, es hora de probar algún código. **Hay dos maneras de usar Python en IDLE**, ¡vamos a verlas!

Lo que aprenderás

- Cómo programar en Python en tu ordenador.
- Utilizar la ventana de Shell de IDLE para ejecutar el código que escribimos en Python.
- Utilizar la ventana de código de IDLE para crear programas más avanzados.

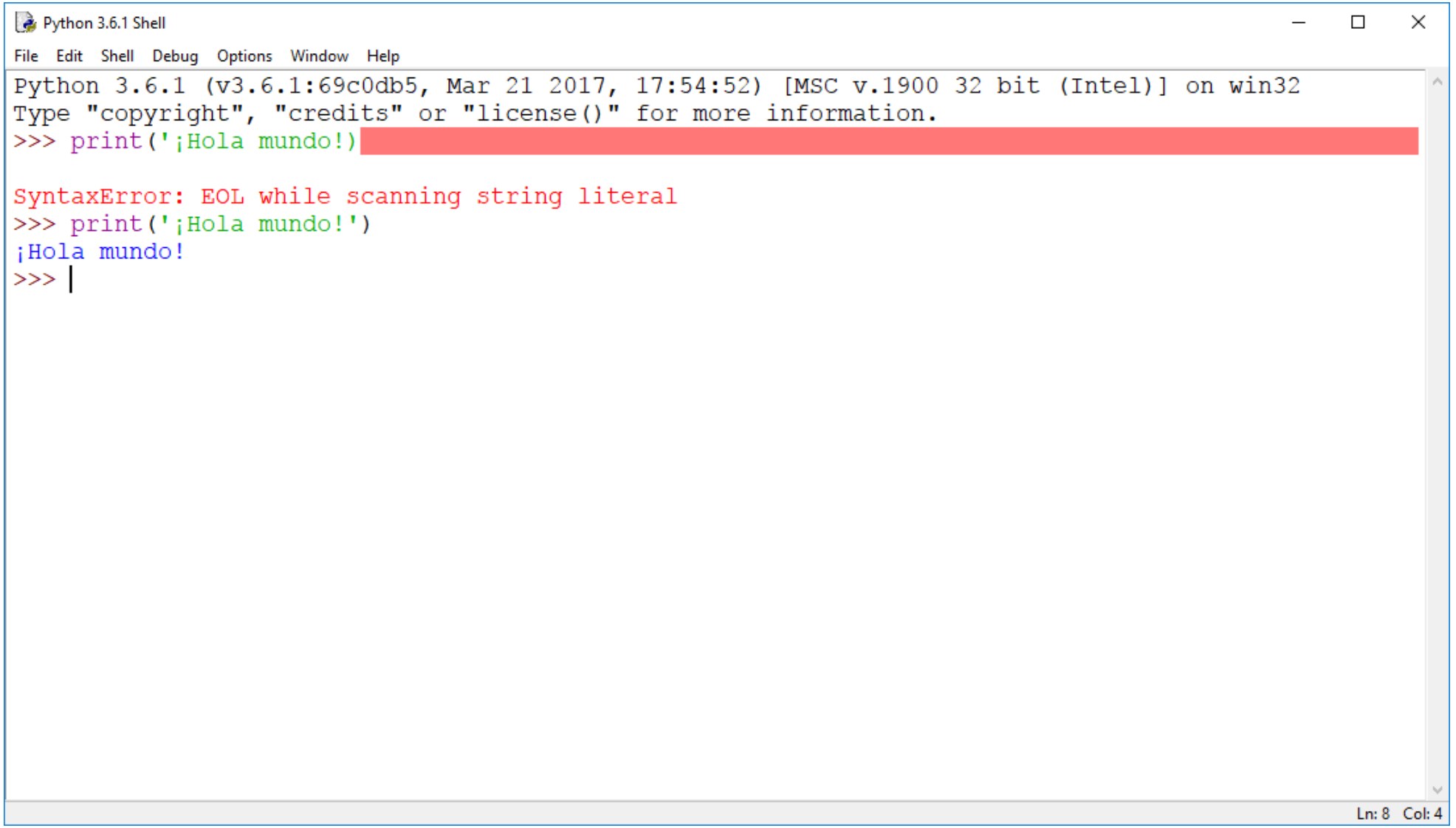
UTILIZAR LA SHELL DE PYTHON

La forma más fácil de empezar con Python es utilizando la ventana de programación que proporciona el entorno IDLE. Se escribe directamente en Python y se ejecuta de inmediato. Cualquier salida o error se muestra por pantalla para identificar los fallos.

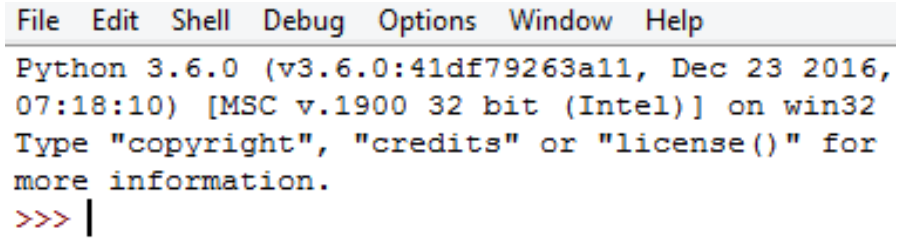


¡Vamos a probar!

Python y IDLE



1. Ejecuta el IDLE de Python. Lo primero que verás será la ventana (Shell) donde empezar a introducir el código.
El prompt (>>>) es la zona de la Shell donde escribirás los comandos para que Python los ejecute.



Aquí se introducen los comandos de Python

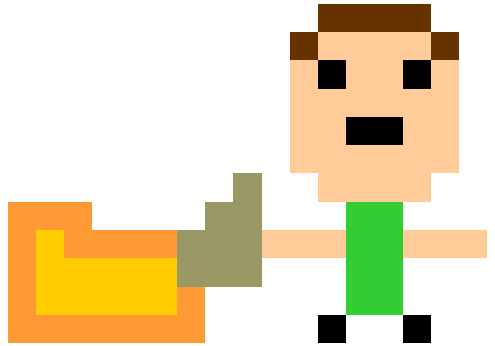
2. La ventana (Shell) de Python muestra la salida de cualquier dato que produce el programa y a la misma vez notifica cualquier error. Si nos hemos equivocado al escribir el código aparecerá un mensaje en rojo y deberás revisar tu código ¿Recuerdas la pantalla anterior?

```
>>> print('¡Hola Mundo!')
¡Hola Mundo!
>>> 3+4
7
>>>
```

Al lado de >>> vemos el código que escribimos, y en azul el resultado de ejecutarlo.

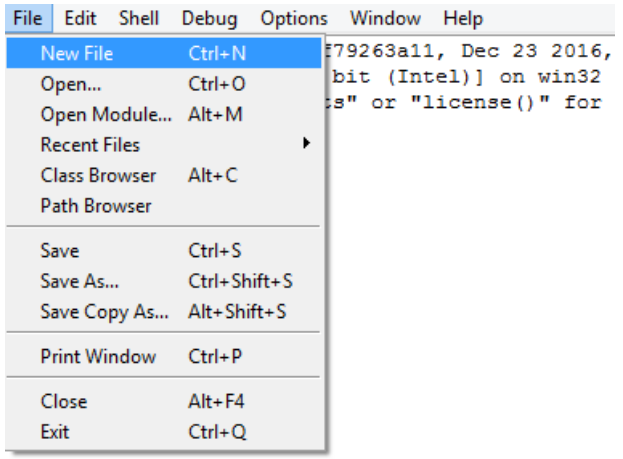
ESCRIBIR CODIGO EN IDLE

Con programas más complicados, es más fácil escribir el código completo y luego pedirle a Python que ejecute el programa. Para ello, utilizaremos otra ventana para introducir el código, que llamaremos "ventana de código".



Código → Guardar → Ejecutar

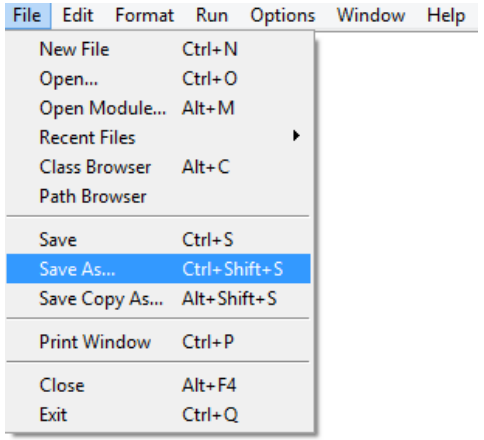
1. Inicia IDLE y se abrirá la ventana (shell). Haz clic en el menú Archivo en la parte superior y seleccione Nueva ventana.



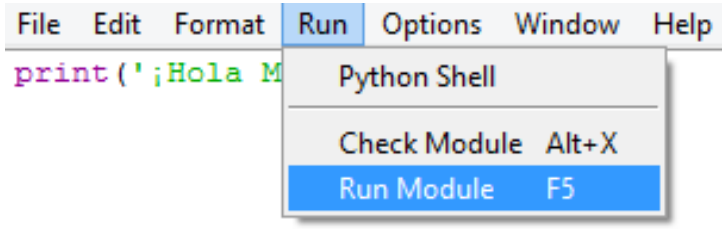
2. Verás que se abrirá otro tipo de ventana llamada ventana de código. Introduce un código en esta ventana. Por ejemplo:

```
print('¡Hola Mundo!')
```

3. A continuación, debes guardar este código haciendo clic en Archivo y a continuación en "Guardar como".



4. Ya has guardado el código y puedes ejecutarlo. Haz clic en el menú Run (Ejecutar) y selecciona Run Module (Ejecutar módulo). Si no has guardado el programa previamente, IDLE te recordará que lo hagas.



5. La salida de datos aparece en la ventana (Shell) y no en la ventana de código. Cualquier resultado se muestra por la ventana (Shell) de Python. ¡Fantástico! ¡Ya has creado tu primer código y lo has ejecutado!

```
>>>
= RESTART: C:/Users/Dmanas/AppData/Loc
al/Programs/Python/Python36-32/uno.py
=
¡Hola Mundo!
>>>
```

6. Es posible que aparezcan errores o mensajes en la ventana (Shell). Si es así, vuelve a verificar tu código e inténtalo de nuevo.

4. PROGRAMAS DE ROBOTS

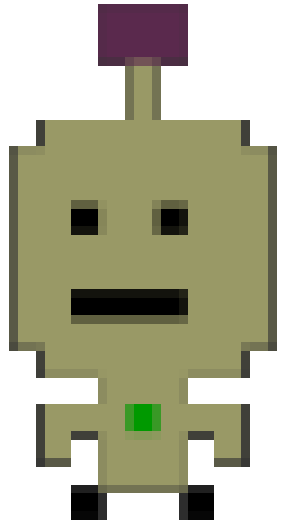
Los ordenadores pueden llevar a cabo tareas complejas, pero no son muy inteligentes por si solos. Así que cada tarea tiene que ser desglosada en una serie de pasos muy simples. Cuando estos pasos se convierten en un lenguaje que una computadora puede entender, se genera un programa de ordenador.

Lo que aprenderás

- Cómo piensan los ordenadores
- Cómo desglosar un problema en pasos más sencillos.
- Combinar instrucciones sencillas y tareas complejas.
- Qué es un algoritmo.
- Qué es un programa de ordenador.

PENSAR COMO UN ORDENADOR

Antes de empezar a programar un ordenador, necesitas aprender a pensar como tal. Esto significa dividir las tareas, enfocándolas a la metodología paso a paso, en secuencias simples. Para conseguir esta forma de pensar, imagina que estás controlando un robot. La tarea es conseguir con este robot, la resolución de una serie de laberintos simples. Si sólo dices "¡Resuelve el laberinto!" al robot, no sucederá nada. Las únicas instrucciones que el robot entiende son:



F = Avanzar al cuadrado en la dirección que está apuntando. Forward()

R = Girar 90 grados (un cuarto de vuelta) a la derecha sin ir a ninguna parte. Right()

L = Girar 90 grados (un cuarto de vuelta) a la izquierda sin ir a ninguna parte. Left()

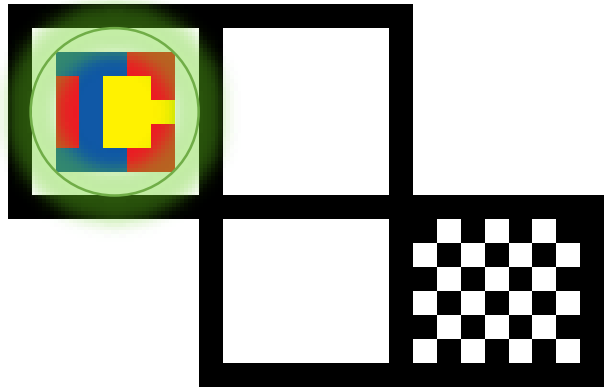
Seguidamente verás un claro ejemplo.

Instrucciones:

Cada instrucción que le damos al robot, significa un paso simple dentro del programa. **El programa completo se forma por una secuencia de instrucciones.**

En Python puedes programar el movimiento de un robot únicamente con estas tres sencillas instrucciones.

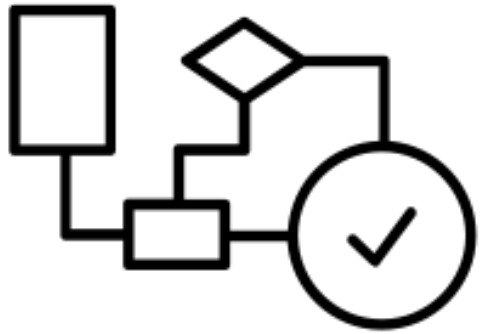
El programa quedaría así: F, R, F, L, F.



El robot comienza en el círculo verde y termina en la bandera a cuadros.

¡Algoritmos asombrosos!

Un algoritmo es un conjunto de pasos que, cuando se siguen en orden, llevan a cabo una tarea. Por ejemplo, la receta para hacer un pastel es un algoritmo. Un programa de ordenador es un algoritmo escrito en una lenguaje que un ordenador puede entender, de modo que puede realizar una tarea en particular.



La secuencia de pasos que hemos descrito para el robot antes, es un algoritmo.

5. PYTHON Y SU TORTUGA

Ya has visto que un programa de ordenador tiene que ser un conjunto de pasos precisos. Ahora, aprenderás cómo dibujar en la pantalla manejando robots, mediante el uso de Python.

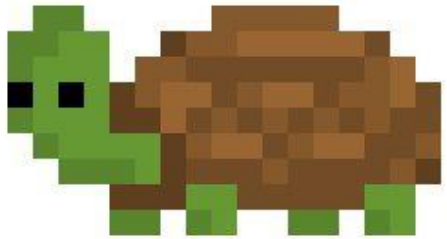
¡Vas a conocer la tortuga de Python!

Lo que aprenderás

- Qué es la tortuga de Python
- Cómo manejar un robot con comandos simples
- Cómo dibujar caminos con la tortuga de Python
- Mover la tortuga de Python en el ordenador

LOS COMANDOS DEL ROBOT TORTUGA

Puedes controlar una tortuga robot escribiendo comandos de Python como:



- forward (50): avanzará 50 pasos en línea recta.
- right (90): girará 90 grados hacia la derecha.
- left (90): girará 90 grados hacia la izquierda.

Cada comando tiene un nombre seguido por un valor entre paréntesis que da información adicional sobre lo que el robot debe hacer, como mover una distancia específica o girar un cierto número de grados. Puedes combinar estos comandos de Python para dibujar prácticamente cualquier cosa.

DIBUJAR UN CUADRADO

Puedes utilizar los comandos de la tortuga para dibujar un cuadrado perfecto.

A continuación verás paso a paso cómo se ha de indicar a la tortuga que siga las instrucciones. A medida que vayas escribiendo los comandos en Python, se irán coloreando para así dejar más legible el código.

Lo primero es activar la tortuga en Python, y luego indicarle cómo debe avanzar para dibujarla.

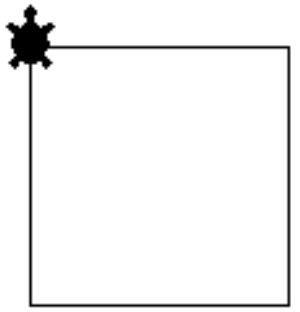
1. Activar la tortuga en la posición inicial.

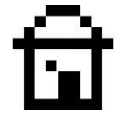
2. La tortuga se mueve 100 pasos en línea recta. Una vez se ha desplazado, la tortuga queda a la espera de recibir la siguiente instrucción.

3. La tortuga gira 90 grados hacia la derecha. Después avanzar efectúa un giro y espera a recibir la siguiente orden.

4. Luego deberemos repetir estos pasos para que vaya dibujando el cuadrado. Mira el código a la derecha.

```
from turtle import*
shape('turtle')
forward(100)
right(90)
forward(100)
right(90)
forward(100)
right(90)
forward(100)
```



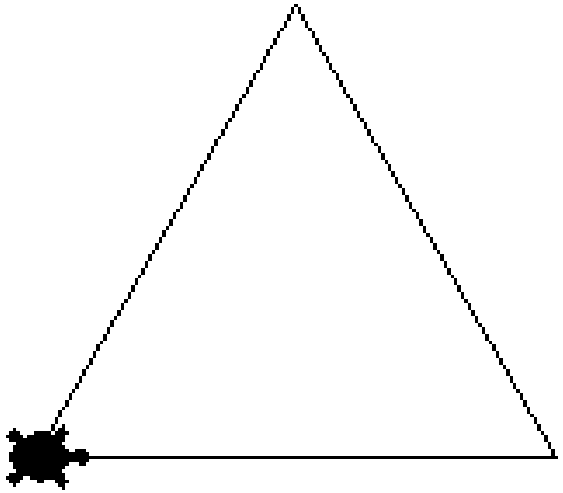


Puedes indicar las distancias que quieres que recorra la tortuga, así como también los ángulos de sus giros, con el fin de poder dibujar cualquier cosa. ¿Dibujamos un triángulo?

Es fundamental que entiendas el sentido y el valor del ángulo de los giros. ¿Por qué 120º?

¿Quieres borrar el espacio de dibujo?
Prueba con el comando reset()

```
>>> forward(150)
>>> left(120)
>>> forward(150)
>>> left(120)
>>> forward(150)
>>> left(120)
```



6. VARIABLES

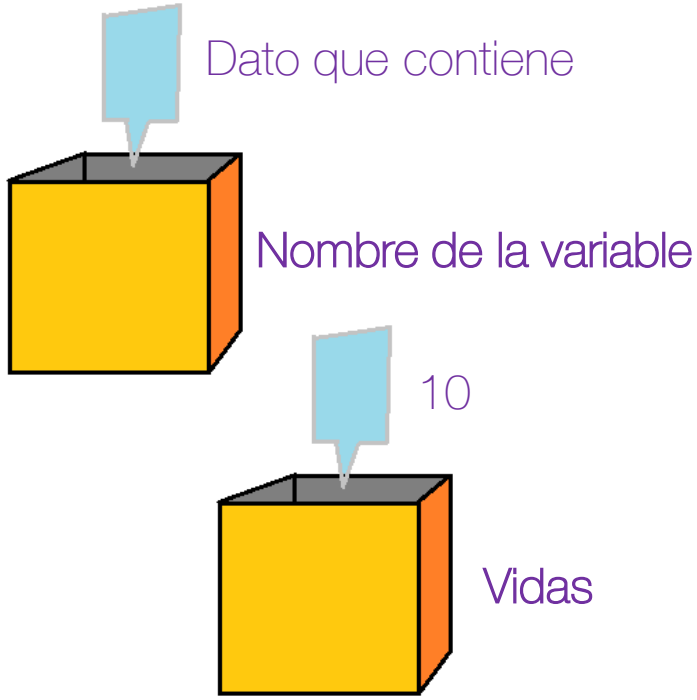
La información, o datos, son el alma de los ordenadores. Los datos se pueden comprender de muchas formas, tales como números, palabras, imágenes y música. Las variables son una manera práctica de almacenar datos.

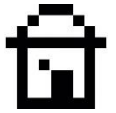
Lo que aprenderás

- Qué es una variable.
- Cómo crear una nueva variable.
- Cómo modificar o copiar variables.
- Cómo mostrar por pantalla los valores de las variables.

VARIABLES – CAJONES DE DATOS

Un programa de ordenador a menudo tiene que almacenar los datos que necesitará más tarde. Para ello se utilizan las variables. Una variable es como una caja con una etiqueta a la que le damos un nombre. Puedes almacenar datos en dicha caja y encontrarlos más tarde usando el nombre que le has dado.





CREAR Y VER EL CONTENIDO DE UNA VARIABLE

Para crear una nueva variable en Python usamos el operador de asignación (=).
A la derecha tienes una variable a la que se le ha dado el nombre de edad y contiene el valor numérico 10.

```
edad = 10
```

Para ver en pantalla lo que hay en una variable usamos el comando print().

```
print(edad)
```

PONER NOMBRE A LAS VARIABLES

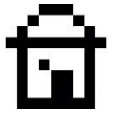
- Trata de elegir nombres que describan los datos que se almacenan en las variables.
- Los nombres de variables pueden usar letras y números, aunque no pueden almacenarse con números.
- También pueden usar subrayado (_), pero no se permiten espacios u otros símbolos.
- Hay algunas palabras que Python no te permitirá utilizar porque ya los está utilizando, como for o print.
- Ten en cuenta que: FLORES es una variable diferente de las Flores y FloReS. Se distingue entre mayúsculas y minúsculas.

MODIFICAR EL VALOR DE LAS VARIABLES

Puedes utilizar el operador de asignación (=) para cambiar el valor almacenado en una variable.

A la derecha puedes ver una nueva asignación que reemplaza el valor anterior (10) por el nuevo valor (11) haciendo que el valor antiguo se substituya. El comando print debería devolver el valor 11.

```
edad = 11
print(edad)
```



COPIAR EL VALOR DE LAS VARIABLES

Puedes copiar el valor de una variable a otra .

La variable edad_ant ahora tiene almacenado el valor edad. No hay ninguna conexión entre las variables, así que si cambias el valor de la edad otra vez, edad_ant mantiene el valor que le asignamos por primera vez.

El valor de edad_ant no cambiará hasta que nosotros asignamos otro valor.

```
edad_ant = edad
print(edad_ant, edad)
```

La salida por pantalla será: 11 11

```
edad = 12
print(edad_ant, edad)
```

La salida por pantalla será: 11 12

TIPOS DE DATOS EN PYTHON

Cada elemento de datos utilizado en Python puede tener ciertas particularidades, siendo un tipo de dato diferente a los demás.

Estos son los tipos de datos más sencillos:

- Números enteros (tipo: int) son números sin punto decimal. Recuerda que usamos puntos y no comas para los decimales en Python. Ejemplos: 3, 10, -2.
- Números punto flotante (tipo: float) son números con un punto decimal. Ejemplos: 3.1, 9.63, -4.1172, 8.5.
- Las cadenas (tipo: str) son secuencias de caracteres limitadas por comillas. Ejemplos: 'Hola', ‘Manolo‘.

7. UTILIZANDO NUMEROS

Los ordenadores son muy buenos trabajando con números. Python puede hacer todas las matemáticas que sabes ¡y mucho más!

Lo que aprenderás

- Qué son las expresiones y los operadores.
- Cómo hacer sumas en Python.
- Utilizar los paréntesis adecuadamente.
- Qué pasa cuando unimos sumas y variables.

MATEMATICAS Y PROGRAMACION

En programación, una suma se denomina expresión.
Una expresión es una combinación de datos y operadores, y tiene un valor.

Operador

```
>>> 3 + 5
```

Datos Datos

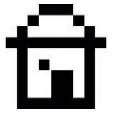
CONOCE LOS OPERADORES

Esta tabla muestra cómo se ven las cuatro operaciones aritméticas más familiares en Python.

Operación	Operador	Expresión de ejemplo	Valor del resultado
Suma	+	13 + 2	15
Resta	-	10 – 8	2
Multiplicación	* (asterisco y no x)	4 * 5	20
División	/	20 / 4	5.0 (siempre obtenemos decimal en la división)

QUE TIPO DE DATOS RESPONDEN

Python tiene reglas para trabajar con los distintos tipos de números: float (decimales) y números int (enteros). Si un cálculo contiene cualquier número de tipo float , o bien si es una división, la respuesta siempre será un decimal. Si todos los números del cálculo son de tipo int y no hay divisiones, la respuesta siempre será un int.



EVITAR ERRORES CON LOS PARENTESIS

Si le dices a Python que calcule 3 + 2 * 8, ¿qué obtendrás? Podría hacer la suma primero: (3 + 2) * 8 = 40. O podría hacer la multiplicación primero: 3 + (2 * 8) = 19.

Cada método da una respuesta diferente, de manera que lo primero es plantear como debes realizar la fórmula y **recordar situar entre paréntesis aquellas operaciones que tienen prioridad.**

Puedes utilizar variables en los cálculos. Digamos que tenemos 12 conejos y que queremos comprar 3 zanahorias para cada conejo, para calcular el número de zanahorias que necesitamos haremos uso de variables.

```
conejos=12
zanahorias=3*conejos
print(zanahorias)
```

Ejecuta tu programa en Python, ¿qué resultado obtienes?

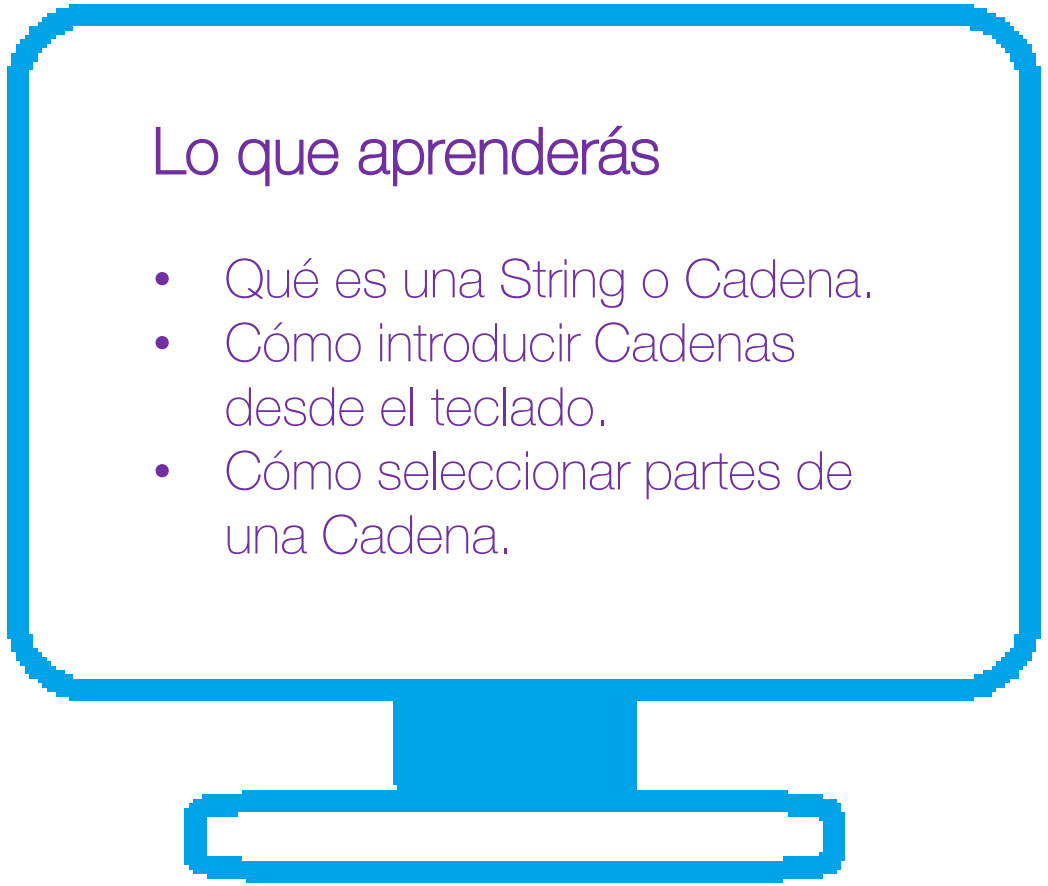
Ten en cuenta que la variable zanahorias solo almacena el resultado de la multiplicación, no la multiplicación en sí. En caso de cambiar el valor de la variable conejos, cambiará también el resultado de la variable zanahorias.

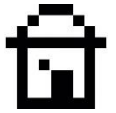
8. CADENAS Y ENTRADAS

Los ordenadores no sólo tratan con números, sino también con letras, palabras y símbolos.
Los programadores llaman a estas **cadenas de caracteres**: "strings".

Lo que aprenderás

- Qué es una String o Cadena.
- Cómo introducir Cadenas desde el teclado.
- Cómo seleccionar partes de una Cadena.





QUE ES UNA STRING

Le puedes decir a Python que algo es una string, o en castellano cadena de caracteres, cuando colocamos dentro de una variable comillas simples. En Python, las cadenas son el tipo de datos str.

Puedes darle a una variable una cadena de caracteres como un valor. En este ejemplo, el valor de la variable llamada "nombre" tiene el valor Manolo.

Algunos ejemplos de variables tipo String:

```
'hola'
'Manolo'
'¡El 7 es el numero magico!'
'G77mk%k$3'
'manolo@correo.com'
```

```
nombre='Manolo'
print(nombre)
```

JUNTANDO STRINGS

Puedes agregar cadenas de caracteres unidas entre si, siempre que la String donde los vayas a almacenar sea lo suficiente grande como para incluir la suma de dos cadenas de caracteres.

```
nombre='Manolo'
saludo='Buenos dias ' + nombre
print(saludo)
```

¿Te has fijado que al final de 'Buenos días' hay un espacio? ¿Sabes por qué?

OBTENER STRINGS DESDE EL TECLADO

Python tiene un comando para poder entrar datos “input()”, que lee lo que se escribe en el teclado y lo guarda como una cadena o string.
Este comando muestra la pregunta en la pantalla (el mensaje) y luego espera a que se escriba una respuesta. Una vez que se pulsa la tecla “ENTER", la cadena se almacena en la variable indicada (comida).

```
nombre='Manolo'
comida=input('¿Qué quieres comer? ')
print(nombre, 'quiere comer', comida)
```

BITS DE LAS STRINGS

Python numera cada carácter en una cadena, empezando desde cero.

Puedes seleccionar algunos caracteres de la cadena de manera individual.

También podemos seleccionar intervalos.

```
correr='¡Corre ya!'
        0123456789
```

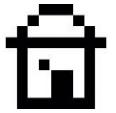
```
correr='¡Corre ya!'
print(correr[0], correr[5], correr[9])
```

Resultado: ¡ e !

```
print(correr[4:8])
```

Resultado: re y

Fíjate que el último valor del rango no aparece, siempre muestra el valor anterior.



9. BUCLES

¿Quieres saber cómo decirle a tu ordenador que haga algo una y otra vez? Para ello deberás aprender como funcionan los bucles en Python.

Lo que aprenderás

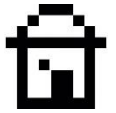
- Qué son los bucles.
- Aprender que son los bucles **for**
- Cómo utilizar los bucles con rango y mostrar los resultados por pantalla.

LISTADO DE NUMEROS

A menudo los ordenadores se utilizan para hacer la misma cosa más de una vez. Imagina que queremos imprimir una serie de números de carreras seguidos.

```
print('Número de carrera: 1')
print('Número de carrera: 2')
print('Número de carrera: 3')
```

Este código con 3 números funciona, pero ¿te imaginas tener que teclearlo para 100.000 carreras? Cansado, ¿no? Además seguro que cometeríamos errores. Por suerte existen los bucles.



```
for numero in range(3):
    print('Número de carrera: ', numero)
```

En la primera parte del bucle:

- con for se llama a la función de tipo bucle
- numero es la variable del bucle que lleva la cuenta de las veces que se ejecuta la instrucción.
- (3) define el número de veces que se ejecuta la instrucción en el bucle.

En la segunda parte del bucle:

- print es la instrucción que se va a repetir en cada vuelta del bucle.
- numero es la variable que irá mostrando en pantalla el valor de cada vuelta.

Esta segunda parte que comienza con print, que como ves se ha tabulado en el código, indica que es la parte del código que se va a ir repitiendo en cada vuelta del bucle. Con cada ejecución la variable numero aumenta en un valor hasta 3 veces, y va cambiando el número que aparece impreso en pantalla.
Fíjate que el primer número que sale es el 0.

¡Cambia 3 por 1000 y mira lo que ocurre en tu programa!

RANGOS: INICIO, FINAL Y PASOS

Puedes usar los números en un rango de bucle for(), para cambiar cuántas veces se repite el bucle y cuantos pasos dará por cada vuelta. Los datos que se contemplan en los bucles, pueden ir cambiando a medida que se realizan las vueltas del bucle.

Todos los bucles deben saber por donde empezar, donde acabar y cuantos saltos deben dar por cada vuelta

Variable del bucle inicio final saltos

```
for contador in range(start, end, steps):
    print(contador)
```

Puedes utilizar un rango concreto de números y decirle exactamente cuántos saltos debe dar dentro de dicho rango. Para ello utiliza el comando for() con la especificación del rango que vas a contemplar range(). Algunos ejemplos:

```
for contador in range(1, 6):
    print(contador)
```

Salida por pantalla: 1 2 3 4 5

```
for contador in range(2, 11, 2):
    print(contador)
```

Salida por pantalla: 2 4 6 8 10

```
for contador in range(5, 0, -1):
    print(contador)
```

Salida por pantalla: 5 4 3 2 1

En el range(), se supone que el paso es 1 si no se especifica un número. También puedes especificar el rango con un único numero range(10), de manera que sería igual que decir range(0, 10).

Después de que el bucle se ha ejecutado el número especificado de veces, el programa continúa trabajando, ejecutando el código que hay a continuación del bucle.

10. MOSTRAR POR PANTALLA

Ya vistes que puedes usar print() para mostrar datos por pantalla. Hay muchas otras maneras útiles de utilizar el comando print().

Lo que aprenderás

- Más usos para el comando print()
- Cambiar la impresión con (separadores)
- Cómo hacer una lista con las salidas de datos por pantalla.
- Cómo imprimir caracteres especiales.

HOLA MUNDO!

Ya sabes imprimir con print() datos del tipo Strings,

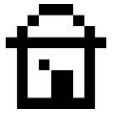
```
print('¡Hola mundo!')
```

Imprimir el valor de variables numéricas

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

Imprimir más de un elemento usando comas

```
print('a tiene el valor' a)
```



SEPARADORES Y FINAL DE LINEA

Cuando envía más de un elemento para imprimir, como por ejemplo print(a, b, c), Python coloca un espacio llamado separador entre cada elemento.

Ahora vas a cambiar los separadores enviando una cadena separadora diferente a la función de impresión, usando sep = '/' dentro de los corchetes del comando print()

```
a = 1; b = 2; c = 3;
print(a, b, c)
```

Se mostrará 1 2 3 (con espacio entre los números)

```
print(a, b, c, sep='/')
```

Se mostrará 1/2/3 (con separador / en los números)

```
print(a, b, c, sep='')
```

Se mostrará 123, sin espacios entre los números. Fíjate que en el código no hemos puesto nada entre las comillas.

No te tienes que limitar a un solo carácter como separador, puedes usar cualquier carácter o cadena de caracteres, incluyendo comas y dos puntos.

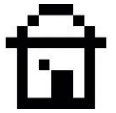
```
a=1
b=2
c=3
print(a,b,c, sep=' más ')
```

Tendrá como resultado `1 más 2 más 3`

Normalmente, print() comienza una nueva línea después de que se haya impreso lo que se le ha pedido. Puedes cambiar esto poniendo end = ‘ ‘ dentro de los paréntesis. Esto le dice a print() que coloque un espacio al final de la impresión en lugar de iniciar una nueva línea. Puedes usar esto en un bucle for para imprimir toda su salida en una misma línea. Puedes poner lo que quieras entre las comillas.

```
for n in range(1, 6):
        print(n, end=' ')
```

Se mostrará 1 2 3 4 5 en la misma línea



TRUCOS PARA LAS STRINGS

Algunas cadenas pueden causar problemas en Python - por ejemplo, ¿qué pasa si quieres poner un apóstrofe dentro de una cadena?

Para evitar confundir a Python usando tres comillas simples, ponemos un carácter de código especial antes del apóstrofe. Este tipo de carácter se llama carácter de escape.

```
print('I'm very happy.')
```

Las apóstrofes en una frase confunden la interpretación de Python. Esto se debe resolver marcando que el carácter de comilla simple se debe escribir como tal y no interpretarse como final de línea.

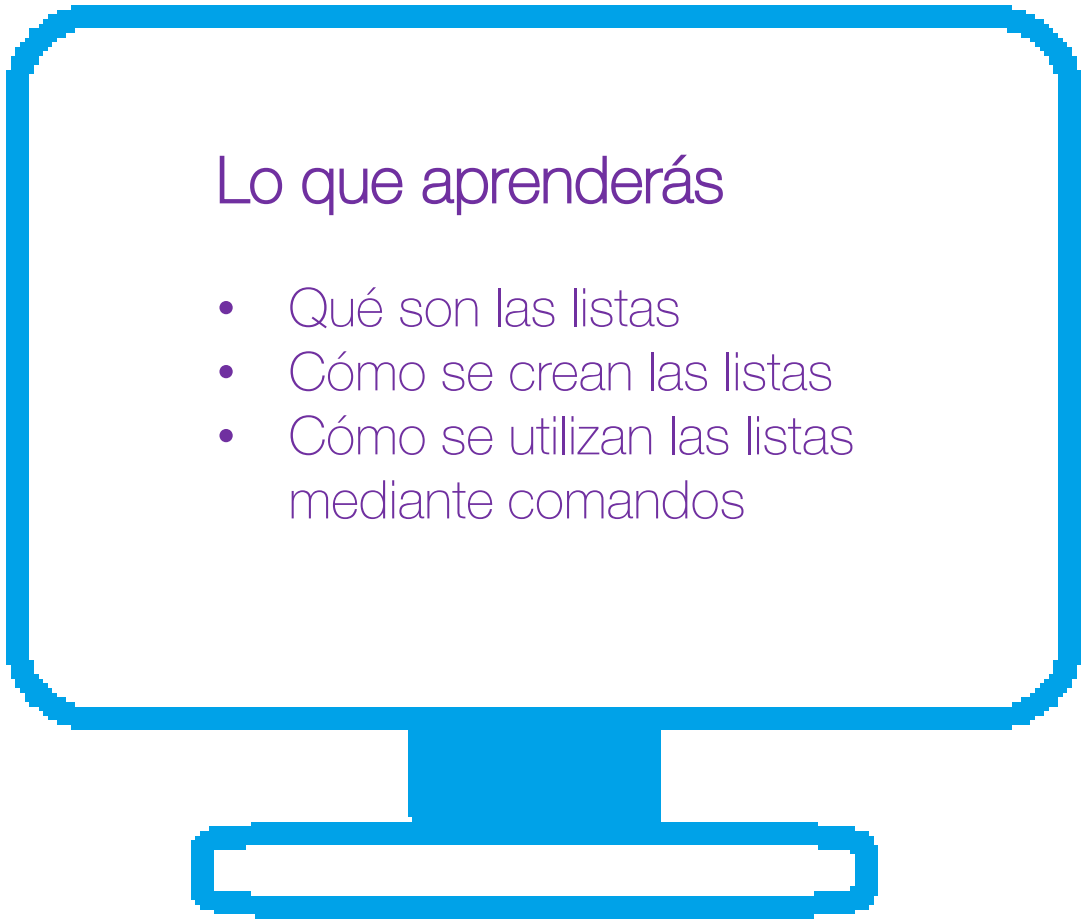
```
print('I\'m very happy.')
```

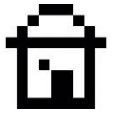
Otra combinación útil es \n, que inserta una nueva línea en el punto en el cual se escribe esta combinación de caracteres.

```
print('hello\nGoodbye')
```

11. LISTAS

Es muy útil poder almacenar montones de información en un mismo lugar. Python utiliza un tipo de dato llamado "lista" para hacer esto.





CREAR UNA LISTA SIMPLE

Una lista puede ser almacenada en una variable, acceder y modificar su contenido. Empieza por crear una lista de cosas que necesitamos comprar

Estas listas que contienen cadenas de caracteres se van a almacenar en una variable llamada compras. Una lista mantiene los artículos en orden.

```
compras = ['cerezas', 'queso', 'helado']

print(compras)
```

La salida por pantalla mostraría:
cerezas
queso
helado

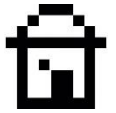
AGREGAR UN ELEMENTO AL FINAL DE LA LISTA

Podemos usar append para agregar un nuevo elemento al final de la lista ya existente de manera muy sencilla.

```
compras.append('banana')
print(compras)
```

Python da a cada elemento de la lista un número de índice para identificarlo. Los números de índice empiezan siempre por cero e indican el orden de la palabra en la lista, lo que se conoce cómo índice numerado.

```
        ['cerezas', 'queso', 'helado', 'banana']
index:      0          1         2          3
```



TRABAJAR CON LISTAS

Puedes usar números de índice para cambiar, leer o eliminar un elemento de una lista. Para recuperar el artículo plátano de nuestra lista de compras deberás escribir el número del índice que marca este elemento en la lista.

Puedes usar un **bucle** para acceder a los elementos de la lista uno a uno y en orden.

```
print(compras[3])
```

El número 3 indica qué palabra de la lista se va a mostrar

```
for objeto in compras:
    print(objeto)
```

Para mostrar todo los elementos de una lista, se puede hacer mediante el uso de un bucle

Para terminar con las listas vamos a ver otras posibilidades:

- Cambiar un ítem de una lista: con la siguiente instrucción cambiaremos lo que haya en la posición 3 de la lista por el término mandarina.

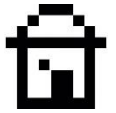
```
compras[3]='mandarina'
```

- Borrar un ítem: borraremos el elemento con posición 3 (recuerda que el primero siempre ocupa la posición 0).

```
del compras[3]
```

- Insertar un nuevo elemento en el orden que queramos: ponemos el elemento naranja en la posición 1.

```
compras.insert(1, 'oranges')
```

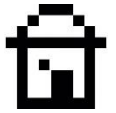


12. CIERTO O FALSO

Python toma decisiones en base a preguntas, llamadas **expresiones booleanas**, con respuestas del tipo sí / no. Los valores que espera recibir o emitir son True o False. Si los traduces al castellano los entenderás como Cierto o Falso.

Lo que aprenderás

- Las expresiones booleanas son **Cierto** o **Falso**
- Los diferentes elementos de comparación que Python utiliza
- Trabajar los operadores de comparación
- Conocer los elementos lógicos **and** y **or**



EXPRESIONES BOOLEANAS

Una **expresión booleana** utiliza un operador de comparación para comparar dos datos.

Por ejemplo, el valor de la expresión que ves a la derecha es True o False: cualquier valor de edad de 17 o inferior hace que la expresión sea verdadera (True). Si la edad es de 18 años o más, será falsa (False).

```
edad < 18
```

Esta expresión comprueba si el dato que hay en la variable edad es menor que 18.

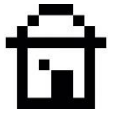
Puedes mostrar por pantalla el valor de una expresión booleana:

True y False tienen su propio tipo de datos, llamado Boolean (su abreviación es Bool). Fíjate que ambos comienzan con una letra mayúscula.

```
print(3 > 2)
print(3 < 2)
```

El resultado es **Cierto,** 3 es mayor que 2

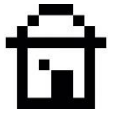
El resultado es **Falso,** ya que 3 no es mayor que 2



OPERADORES DE COMPARACION

Los números pueden ser comparados en Python usando una serie de caracteres especiales u operadores (==) (!=) (>) (<) (<=) (>=)

Operadores de comparación	Expresiones Booleanas	Resultados (con valor 10)
Igual que	edad == 10	Cierto
Más grande que	edad > 10	Falso
Más pequeño que	edad < 10	Falso
Más grande o igual que	edad >= 10	Cierto
Más pequeño o igual que	edad <= 10	Cierto
Diferente a	edad != 10	Falso



COMPARANDO CADENAS

Las cadenas también pueden ser comparadas, pero su contenido debe ser exactamente el mismo para que sean iguales, incluso los espacios y las letras en mayúscula tienen que ser idénticas. Algunos ejemplos:

`'Ana' == 'Ana'`	Cierto. Son dos cadenas exactamente iguales.
`'Ana' == ' Ana'`	Falso. La segunda tiene un espacio.
`'Ana' != 'Eva'`	Cierto. Son completamente distintas.
`'10' == 10`	Falso. La primera es una cadena y la segunda un número.

OPERADORES LOGICOS

Puedes combinar dos expresiones booleanas con los operadores and y or de Python para realizar comparaciones más complejas, incluyendo más de una condición, por ejemplo. Con and se deben cumplir las dos condiciones, con or solo una de las dos.

```
edad > 10 and edad < 18

edad <= 10 or edad >= 18
```

Para que sea cierta, la variable edad ha de tener un valor mayor que 10 y menor que 18.

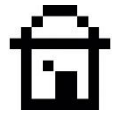
Para que sea cierta, la variable edad ha de tener un valor menor o igual que 10 ó mayor o igual que 18.

13. BIFURCACIONES

Ya vistes cómo tomar decisiones True / False usando expresiones booleanas. Ahora verás cómo usarlas en un programa para elegir las instrucciones que llevará a cabo.

Lo que aprenderás

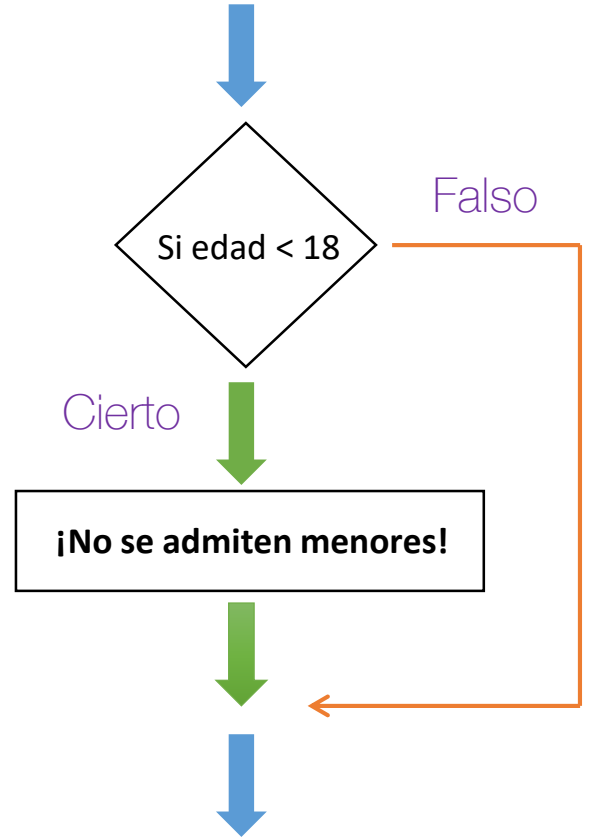
- Qué es una bifurcación en programación
- Cómo convertir los resultados de expresiones boolenas en acciones dentro de un programa.
- Utilizar **if** y **if-else** en Python.



QUE ES UNA BIFURCACION EN PYTHON

Con una bifurcación, se ejecuta un código diferente dependiendo de si una condición (una expresión booleana) es True o False.

En el diagrama de flujo de la derecha puedes ver que el código tiene dos bifurcaciones: una imprime el mensaje “¡No se admiten menores” y la otra salta la impresión del mensaje.



If ("Ejecutar o ignorar")

Las bifurcaciones en Python ejecutan un bloque de código si una expresión booleana es True (verdadera) pero lo omiten si la expresión es False (falsa).
Podemos entender la sentencia If como “si esta condición es True, ejecuta las instrucciones en este bloque de código".

If - else ("hacer esto, sino hacer esto otro")

Esta expresión ejecuta un bloque de código si una expresión booleana es True, pero por el contrario se ejecutará un bloque diferente si es False. Puede entenderse como: "si esta condición es verdadera entonces sigue las instrucciones en este bloque del código, sino haz estas otras instrucciones".

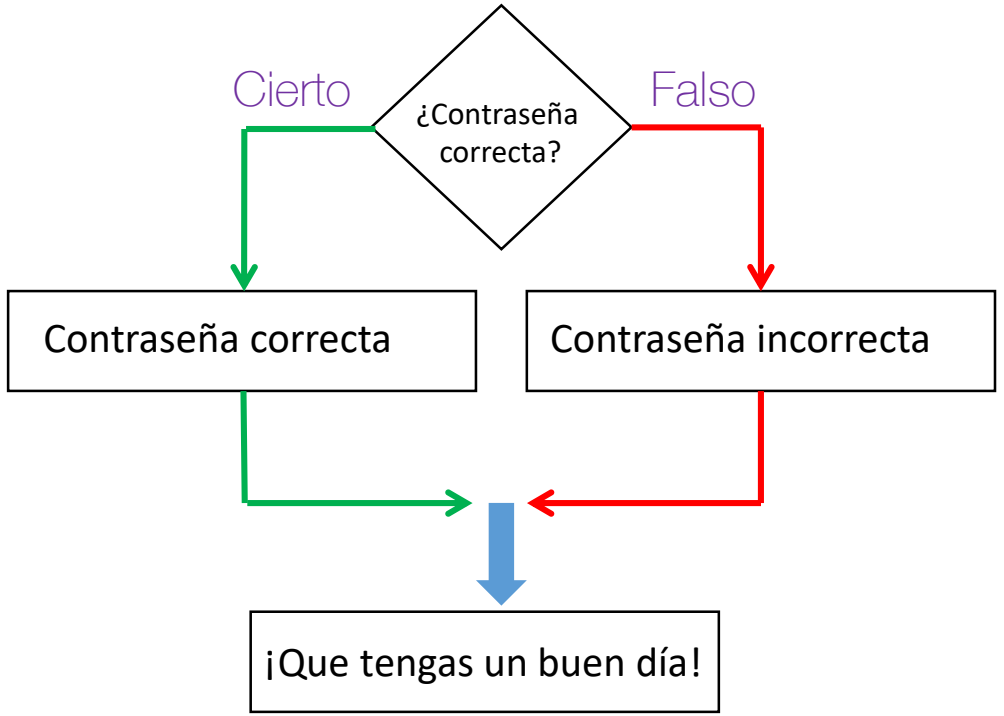
```
contraseña = input('introduce la contraseña: ')
if contraseña == 'swordfish':
    print('Contraseña aceptada')
else:
    print('Alerta intrusos')
print('Tenga un buen día')
```

Veamos que hace este código:

- Se introduce el valor de la variable contraseña por teclado mediante una función input.
- Si la contraseña es la palabra “swordfish” se imprime en pantalla el mensaje “Contraseña aceptada”.
- Si no lo es, aparece en pantalla el mensaje “Alerta intrusos”.
- Una vez aceptada o rechazada la contraseña, se llega a la instrucción que presenta en pantalla el mensaje “Tenga un buen día”.

PRUEBALO TU MISMO

¿Te animas a escribir el código del diagrama de flujo de la derecha? Utiliza la ventana de código. Si lo ejecutas desde la línea de comandos en la ventana (shell), es probable que sigas cometiendo errores y se convertirá en frustrante… y no queremos que lo pases mal. ☺

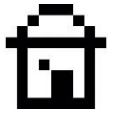


14. BUCLES DEL TIPO WHILE

Los bucles que hemos visto hasta ahora son útiles si sabes cuántas veces quieres hacer algo. Pero si quieres seguir adelante ejecutando instrucciones hasta que suceda algo específico, necesitas un bucle While.

Lo que aprenderás

- Qué es un bucle tipo While
- Cómo usar los bucles tipo While
- Cómo hacer un bucle infinito
- Cómo salir de un bucle infinito



BUCLES WHILE

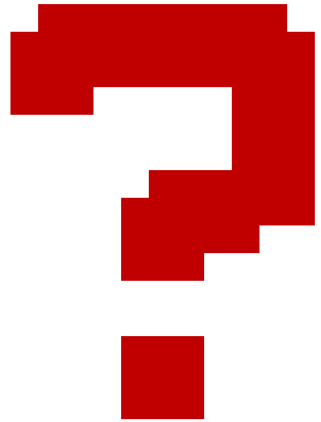
Un bucle While comprueba una condición en la parte superior del bloque de un código y repite el bloque sólo mientras la condición permanezca True.

El Final de bucle o loopends en Python, se dan una vez que las condiciones son False.

Las instrucciones de código situadas dentro del bucle está tabuladas, para así dejar claro que es lo que se produce mientras se cumple la condición.

JUEGO DE ADIVINANZAS

Puedes usar un bucle tipo While para hacer un juego simple, adivinando el nombre de un animal. El bucle While continúa repitiendo, pidiéndole al jugador que escriba su respuesta, hasta que se escriba el nombre correcto del animal. El bucle entonces termina y finaliza el juego.



Ya puedes revisar el código en la pantalla siguiente.

```
print('Adivina el animal')
respuesta = ''
correcto = 'elefante'
while respuesta != correcto:
    respuesta=input('¿De qué animal se trata? ')
print('Lo adivinaste')
```

Veamos que hace este código:

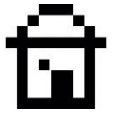
- Se presenta en pantalla el mensaje “Adivina el animal”.
- Se define una variable llamada respuesta que en principio está vacía.
- Se define una variable llamada correcto con el valor “elefante”.
- Se inicia el bucle while con la comparación de si el valor de la variable respuesta es distinto del de la variable correcto.
- Dentro del bucle, y mientras los dos valores sean diferentes, se pedirá que introduzcamos por teclado de qué animal se trata, asignando lo que escribamos a la variable respuesta, para hacer la comparación de nuevo con la variable correcto.
- Cuando finalmente el valor de respuesta no es distinto del de la variable correcto, se sale del bucle y se imprime en pantalla el mensaje “Lo adivinaste”.

CONTANDO

Puedes usar una variable para contar cuántas veces se repite un bucle While. Para aumentar el contador en uno cada vuelta, lo indicarás como: a = a + 1. A veces se ha de escribir esto muchas veces, de manera que puedes simplificar escribiendo a+ = 1.

Vamos a usar este tipo de bucle para escribir un código que nos imprima los números del 1 al 5.

```
a= 0
while a < 5:
    a = a + 1
    print(a)
```



BUCLES INFINITOS

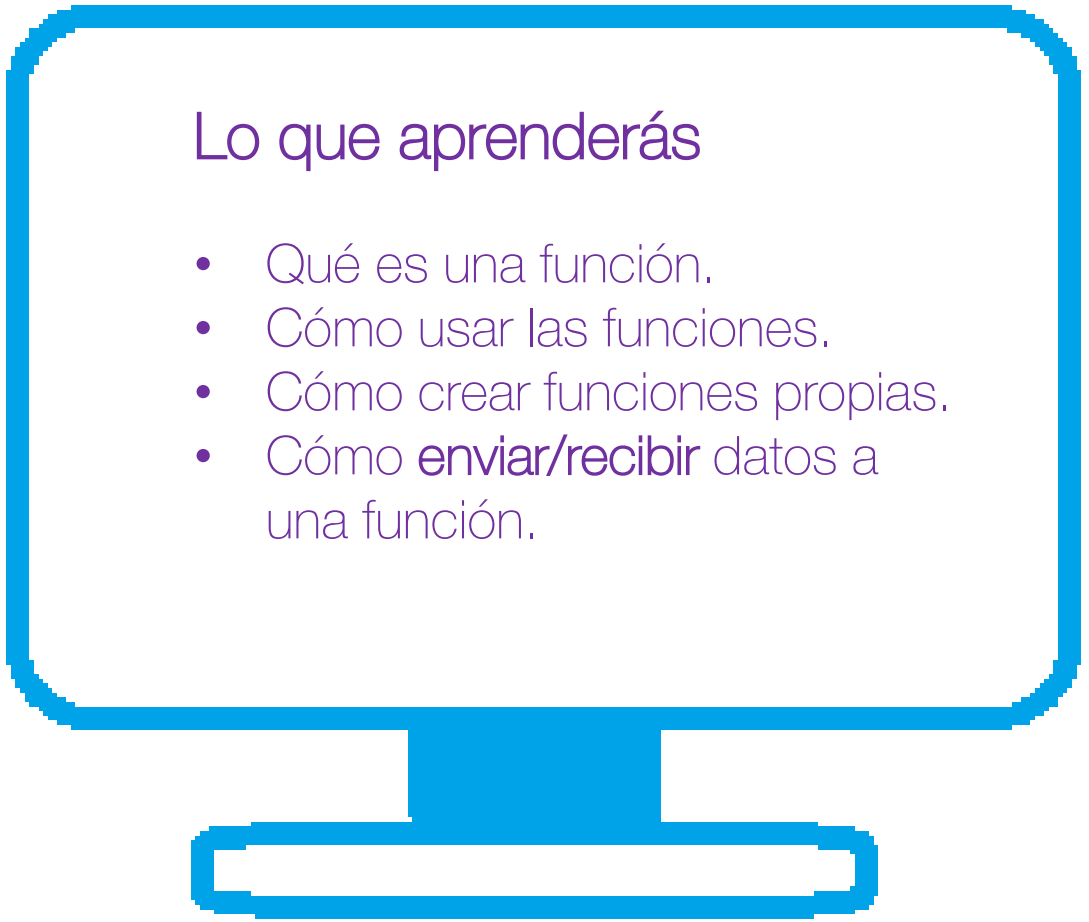
Con un bucle While, puedes incluso hacer bucles que nunca dejan de tener la expresión booleana con el valor True. El valor False nunca ocurre (lo cual haría que termine el bucle).
¡Y de este modo el bucle While continuará para siempre!

```
while True:
    print('Me gusta programar en Python')
```

Para detener la ejecución del bucle y detener la Impresión en pantalla, deberemos pulsar las teclas CTRL y C simultáneamente.

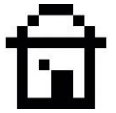
15. FUNCIONES

Seguramente te encontrarás en la situación de que un código es muy útil, de manera que quieres reutilizarlo. Para evitar escribir este código más de una vez, lo puedes convertir en una **"función"** de manera que podrás utilizarlo cuando quieras solo escribiendo el nombre de la función.



QUE ES UNA FUNCION

Una función es **un bloque de código al que se le ha dado un nombre**. Puedes usar ese código en un programa simplemente incluyendo el nombre de la función.
A veces pasamos datos a una función y a veces será la función la que nos devolverá datos.



FUNCIONES QUE YA CONOCES

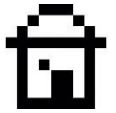
Ya utilizaste algunas de las funciones integradas en Python sin darte cuenta. Todas las órdenes de la tortuga son funciones, tales como forward(100), left(90), etc. Otros ejemplos:

```
print (a, b, c)
```

Print es una función a la que le pasamos datos para que los muestre en pantalla.

```
nombre = input('¿Cuál es tu nombre?')
```

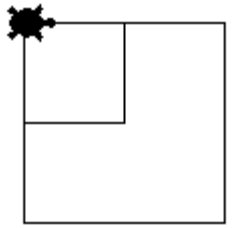
Con la función input damos valor a una variable llamada nombre a partir de lo que respondamos a la pregunta que aparece en pantalla.



CREAR FUNCIONES PROPIAS

Puedes crear y utilizar tus propias funciones. Primero tienes que definir la función usando la palabra clave def para decirle a Python que no ejecute ninguna de las instrucciones. Simplemente almacena el código con el nombre dibuja_cuadrado para ser usado más tarde.

```
def dibuja_cuadrado(medida):
    for n in range(4):
        forward(medida)
        right(90)
dibuja_cuadrado(50)
dibuja_cuadrado(100)
```



Analicemos el código para entenderlo:

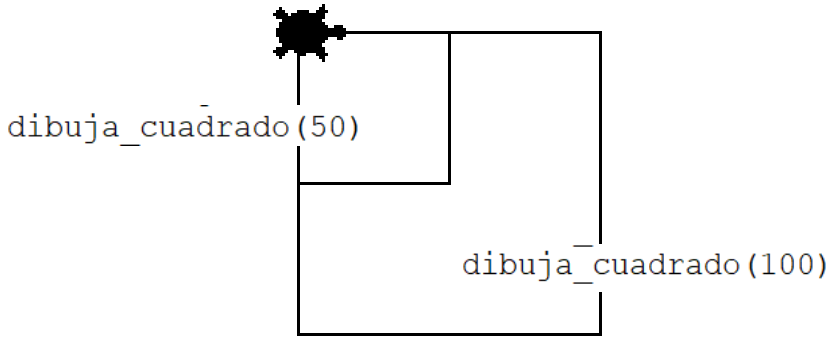
```
def dibuja_cuadrado(medida):
    for n in range(4):
        forward(medida)
        right(90)
dibuja_cuadrado(50)
dibuja_cuadrado(100)
```

- La primera línea define la función “dibuja_cuadrado” que lleva asociada la variable “medida” que es la que le facilita los datos a la función para que haga lo que tiene que hacer (dibujar cuadrados).
- Las 3 líneas siguientes son el código de la función: hay un bucle (for) que repite la acción de pintar hacia delante (forward) los pasos que haya en la variable “medida” y de girar a la derecha 90º (right) en 4 ocasiones.
- Las 2 últimas líneas son las llamadas a la función con dos valores de la variable “medida” distintos: 50 y 100, para que se dibujen dos cuadrados distintos.

Para ejecutar las instrucciones dentro de la función, sólo usas su nombre seguido por el tamaño del cuadrado entre paréntesis.

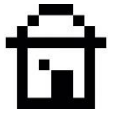
```
dibuja_cuadrado(50)
dibuja_cuadrado(100)
```

Cuando Python ve un nombre de función, simplemente reemplaza el nombre con todo el código de la definición de función y lo ejecuta. El uso de una función también se conoce como llamar a la función.



No olvides incluir en tu código

```
from turtle import*
shape('turtle')
```



OBTENER DATOS DESDE UNA FUNCION

Las funciones pueden devolver datos al código desde el que fueron llamadas, usando el comando return.
La función que ves a la derecha calcula el área de un rectángulo.

Recuerda que la fórmula es:
Área = base x altura

```
def area_rect(base, altura):
    area = base * altura
    return area
print(area_rect(10, 5))
```

La instrucción return devuelve el valor de la variable área al código principal, pudiendo así imprimir por pantalla el valor del área del rectángulo.

¿Te da 50?