Raspbian

APRENDE
Y DIVIÉRTETE

GOBIERNO DE LA CIUDAD DE MÉXICO

SECTEI

CIUDAD **INNOVADORA** Y DE **DERECHOS**

Python

Módulo 4

Semana	Tiempo sugerido	Temas/ Subtemas	Aprendizaje esperados	Eje -Ámbitos-Ambientes Sociales de Aprendizaje
14	70-90 minutos	¿Qué es Python?	Conoce el lenguaje de programación Python. Refuerza sus conocimientos en inglés.	Número, Algebra y Variación. Reforzar concepto de Sistema en Ciencia y Tecnología.
15	70-90 minutos	Elementos básicos: Tu primer programa, tipo de datos.	Interactúa con el lenguaje de programación Python. Desarrolla competencias de análisis matemático.	Sentido numérico. Manejo de la información.
16	70-90 minutos	Solicitar e imprimir datos.	Comprende la estructura de programación en Python. Interactúa con el usuario.	Sentido numérico. Manejo de la información.
17	70-90 minutos	Ciclos y Funciones	Identifica qué es un ciclo. Reconoce las diferencias y usos de for y while Reconoce las reglas para escribir un programa con ciclos.	Sentido numérico. Manejo de la información.
18	90-120 minutos	Turtle	Utiliza el paradigma de Programación Orientada a objetos. Realiza programas con las herramientas aprendidas.	Sentido numérico. Manejo de la información. Forma espacio y medidas.
19	90-120 minutos	Fractal	Reconoce el concepto de iteración, Visualiza un fractal con programación en Python	Sentido numérico. Manejo de la información. Forma espacio y medidas.

14. ¿Qué es Python?

Aprendizajes esperados

Habilidades	Medio	Contenido	Finalidad
Reconoce un lenguaje de programación de alto nivel.	Thonny Python IDE	Introducción ¿Qué es Thonny Python IDE? ¿Cómo usar Thonny Python IDE?	Conocer qué es Python. Reconocer y explorar el entorno de desarrollo de Python para Raspberry PI. Reforzar sus conocimientos en el idioma inglés.

14. ¿Qué es Python?

Introducción

Python es un lenguaje de programación de alto nivel; es decir, expresa los algoritmos de manera entendible para nosotros los humanos y no es tan abstracto como el lenguaje de máquina, o también llamado de bajo nivel; que se comunica sólo con 0s y 1s para emitir una instrucción.

En Python no es necesario compilar ni enlazar con otra interfaz.

El principal objetivo que persigue este lenguaje es la facilidad, tanto de lectura, como de escritura.

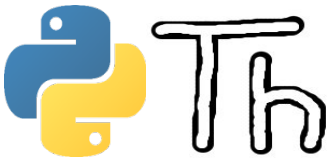
El nombre del lenguaje proviene de la afición de su creador original, Guido van Rossum, por los humoristas británicos Monty Python.

Existen diferentes entornos de desarrollo para programar Python. Las prácticas que elegimos están hechas en: Thonny Python; precargado en la Raspberry Pi; también puedes descargarlo para una computadora de escritorio con Windows, Linux o Mac Os.

¿Qué es Thonny Python IDE?

Un IDE es un Entorno de Desarrollo Integrado Interactivo, *(Integrated Development Environment).*

Es un entorno que permite generar programas computacionales. Dentro de estos IDE´s para Python, se encuentra Thonny Python IDE.



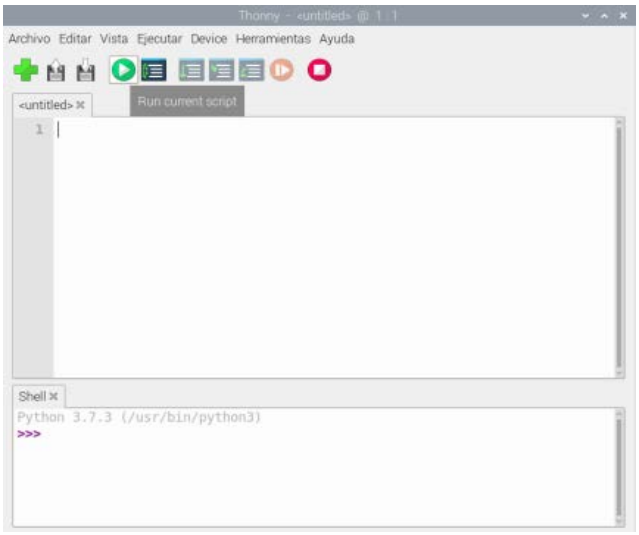
¿Cómo usar Thonny Python IDE?

Para comenzar con Thonny Python IDE, dirígete a la parte superior izquierda y da clic en el ícono de Raspberry. Ve a la opción: **Programación** y selecciona: **Thonny Python IDE**.

Se abre una pantalla con un menú en la parte superior y dos ventanas.

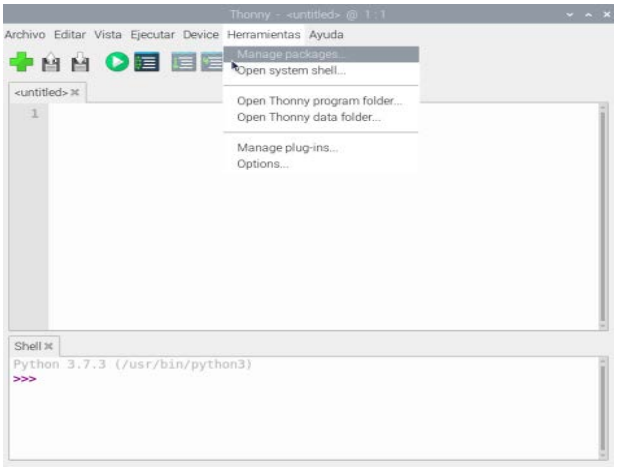
La parte superior tiene el nombre “untitled” (sin título). Puedes cambiarlo, será el nombre con el que guardarás el programa.

La parte inferior se conoce como: Shell (intérprete de comandos) y es donde se ejecutará el programa.

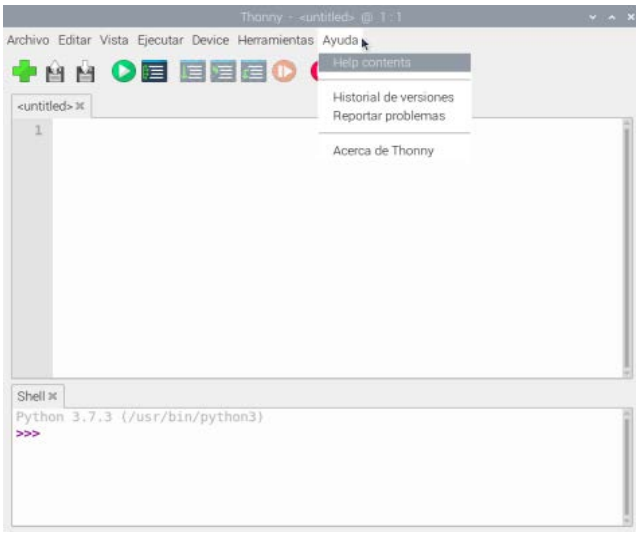


Ahora da clic en **herramientas**, y selecciona la opción "Manage packages". Ahí se encuentran todas las librerías disponibles para utilizar en tus futuros programas.

Hay librerías de todo tipo: matemáticas, herramientas de criptografía (ocultar mensajes y/o descifrar), uso de la cámara, etc.



En la pestaña **Ayuda**, puedes acceder a la página de Thonny Python y navegar por tutoriales y ejemplos de programas.



Actualmente Python es utilizado para el análisis de datos, programación de juegos, aplicaciones de inteligencia artificial, sitios web, etc.

Programar es una actividad colaborativa. Si tienes una idea que quieras desarrollar, busca en foros y/o tutoriales, verás la manera en que programadores han resuelto problemas similares.

Aprender a programar es como aprender a nadar, métete a la alberca, sin miedo y disfruta las posibilidades de materializar lo que imaginas.

Python contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.

Lo que realmente hace especial Python en la Raspberry Pi, es el poder utilizar los pines GPIO para conectar sensores y actuadores. Es decir, interactuar con el mundo físico.

Antes de terminar la lección. Dirígete al Intérprete de comandos (Shell) y utilízala como calculadora. Introduce las operaciones que se encuentran en la siguiente imagen.

Operador	Descripción	Ejemplo
+	Suma	3+2 5
-	Resta	5-4 1
*	Multiplicación	8*3 24
**	Exponente	2**3 8

/	División	10/5 2
//	División Entera	3.5//2 1
%	Módulo	7%2 1

Guido van Rossum es el creador del lenguaje de programación Python. Creció en los Países Bajos y estudió en la Universidad de Amsterdam, donde se graduó con una Maestría en Matemáticas e Informática. Su primer trabajo después de la universidad fue como programador en CWI, donde trabajó en el lenguaje ABC, el sistema operativo distribuido Amoeba y una variedad de proyectos multimedia. Durante este tiempo creó Python como proyecto paralelo. Luego se mudó a los Estados Unidos para trabajar en un laboratorio de investigación sin fines de lucro en Virginia, se casó con un texana, trabajó para varias otras empresas nuevas y se mudó a California. En 2005 se unió a Google, donde obtuvo el rango de Ingeniero Senior de Personal, y en 2013 comenzó a trabajar para Dropbox como Ingeniero Principal. En octubre de 2019 se retiró. Hasta 2018 fue Python ' s BDFL (Benevolent Dictator For Life), y todavía está profundamente involucrado en la comunidad de Python. Guido, su esposa y su hijo adolescente viven en Silicon Valley, donde les encanta caminar, andar en bicicleta y observar aves.

15. Principios básicos

Aprendizajes Esperados

Habilidades	Medio	Contenido	Finalidad
Interactúa con el lenguaje de programación Python. Desarrolla competencias de análisis matemático.	Thonny Python IDE	Tu primer programa. ¿Qué es un tipo de dato? ¿Qué es una variable?	Escribir un primer programa en Python. Reconocer los diferentes tipos de datos que puede utilizar. Reforzar sus conocimientos en el idioma inglés.

15. Principios básicos

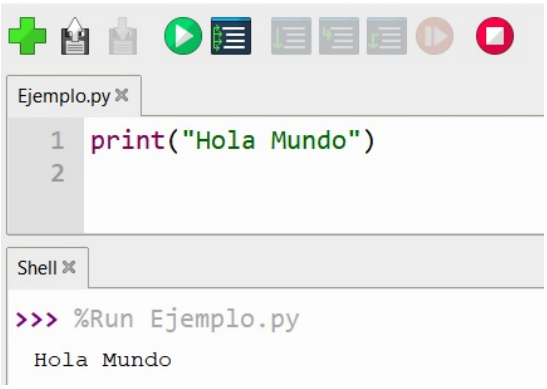
Tu primer Programa

Es una costumbre en el mundo de la programación que el primer programa que se crea es el llamado: "Hola Mundo". Y simplemente es imprimir la frase en un dispositivo de visualización o terminal.

Instrucciones.

¡Abre Tonny Python!

Escribe las instrucciones de la imagen siguiente y después presiona la flecha verde que se encuentra en el menú principal. Thonny IDE te pedirá que pongas un nombre a tu archivo. Ponle “Ejemplo” y ahora sí te imprimirá la frase en el Shell.



Puedes probar cambiando el texto (lo que está en verde y dentro de las comillas). Vuelve a ejecutar el código.

La función que acabas de escribir es: **print**, (imprimir); aquello que esté dentro del paréntesis y en comillas es lo que va a desplegar. Ahora escribe **Print (con P mayúscula)** y ve qué sucede.

Python sí distingue entre mayúsculas y minúsculas, así que sé muy observador y cuidadoso. Ahora, en lugar de: (“Hola Mundo”), escribe: (5+3).

Cada lenguaje tiene sus reglas, lo mismo ocurre con la programación, debes aprender las formas necesarias para que se entienda lo que quieres ejecutar.

Por ejemplo, abajo verás dos versiones de un programa, escrito en lenguajes diferentes: el primero está descrito por bloques, como en Scratch y el segundo está escrito en Python.


```
numero = 5
if numero > 3:
    print("Si se cumple ")
else:
    print("No se cumple ")
```

¿Qué es un tipo de dato?

Un dato es una representación simbólica de la información que recibe o manipula tu lenguaje de programación.

Python es un lenguaje de programación de propósito general y por ello contiene una gran cantidad de tipos de datos con los que se puede programar.

Numéricos: Son simplemente números. En Python los números pueden ser **enteros (int)** (1, 2, 3,...), **reales** o de **punto flotante (float)** (3.1415, 2.36); también puedes operar con números complejos (a+bi). Es decir, números con una parte real y una parte imaginaria. No te preocupes por estos últimos, eso no se verá en este módulo; solo como breviario cultural vale la pena que sepas que un número imaginario se representa como factor de i, que es el operador imaginario.

Al número raíz cuadrada de -1, se le dio el nombre de i. Es decir, i es un número que multiplicado por si mismo dé menos uno; curioso ¿no? Como no hay número real que lo logre, se le llamó imaginario.

Es apasionante descubrir ese tipo de peculiaridades. Si te interesa busca más sobre números imaginarios y complejos.

Booleanos: Son tipos de datos booleanos que se utilizan para representar los valores: verdadero o falso, mediante las palabras reservadas ***True*** **o** ***False*** respectivamente. Este tipo de datos es muy importante para el control de flujo de un programa.

Su nombre proviene de un matemático que generó un algebra para operar con ese tipo de datos. George Boole.

Cadenas: Son secuencias de símbolos (letras o números). Aunque las cadenas no son usualmente importantes para análisis numérico, sí lo son para mostrar resultados por la terminal o Shell. Una cadena debe delimitarse con comillas simples o dobles

Listas: Una lista es una colección de objetos: números o cadenas (letras o números entre comillas). Una lista se delimita utilizando **[]** y sus elementos han de separarse con comas. Es posible acceder a sus elementos indicando el índice del elemento deseado.

Tupla: Es una colección de objetos de distinto tipo. Es decir, son lista de elementos de diferente tipo de dato.

Diccionario: Un diccionario se compone de dos partes: una clave y un valor. La clave y el valor se separan con: (dos puntos) y sus elementos con comas.

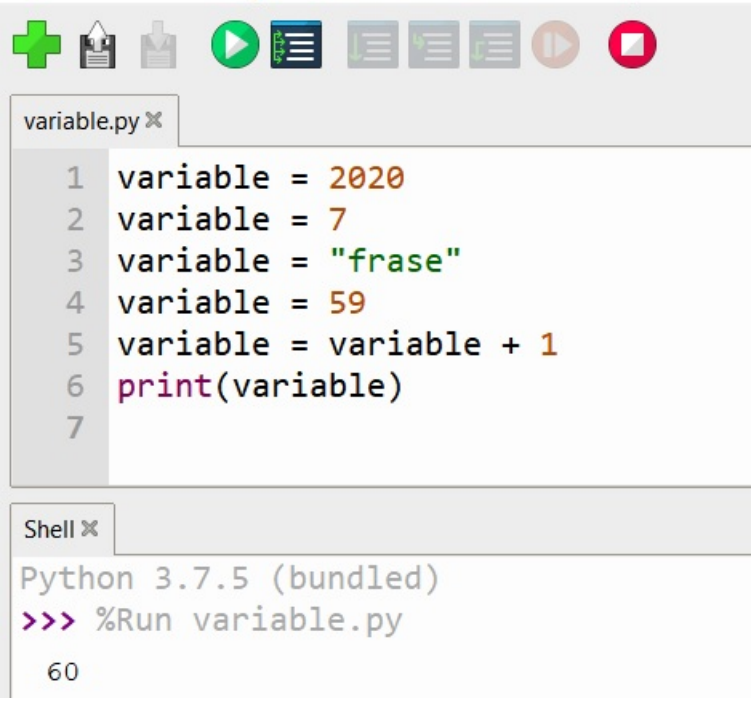
¿Qué es una variable?

Se define como variable al espacio reservado de la memoria que almacena un dato; esta puede cambiar de valor durante el desarrollo del programa.

Los diferentes tipos de datos pueden almacenarse en una misma variable sin especificar de antemano el tipo de datos que almacenará. Esto no ocurre igual en todos los lenguajes de programación.

Ejecutando el siguiente ejemplo, puedes observar que la variable llamada **"variable"** toma diferentes valores en las diferentes líneas del programa.

En la última línea se imprime el valor, que al ser dinámico obtiene el valor de las dos anteriores instrucciones, es decir, 60=(59+1).



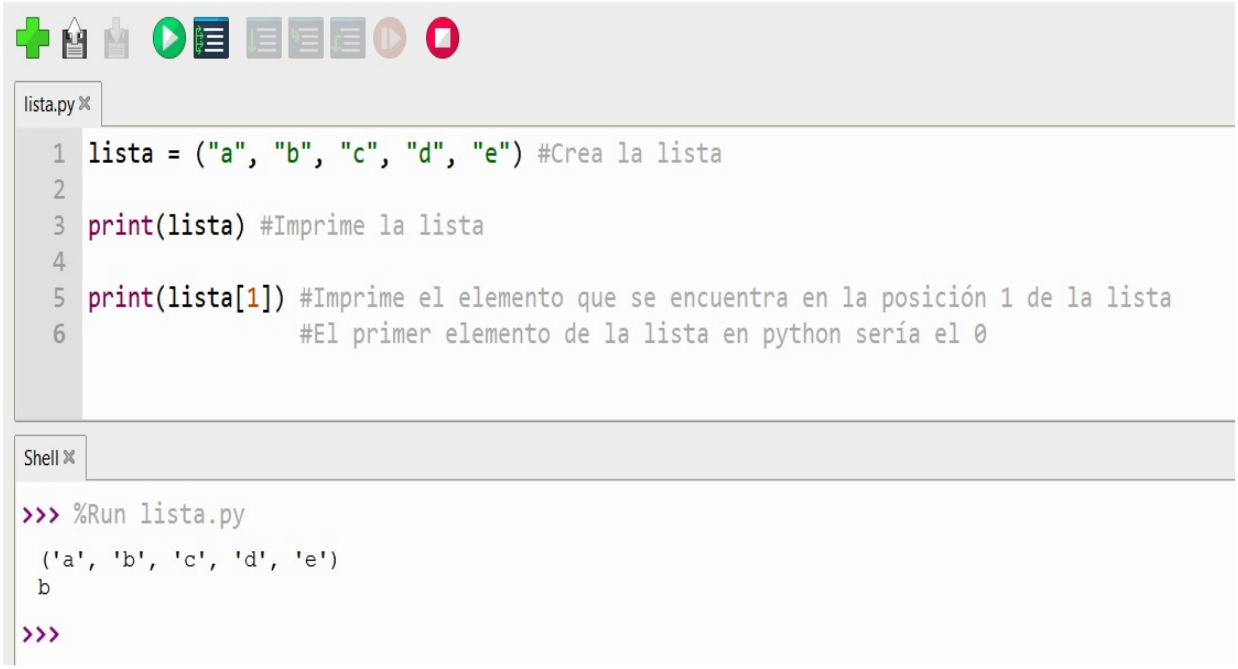
Ejemplo de lista:

La siguiente "lista" contiene los caracteres "a, b, c, d y e".

Escribe las instrucciones de la imagen siguiente. En ellas, se solicita que se imprima la lista completa y luego se pide imprimir el segundo elemento de la lista.

Cuando escribes la instrucción **print(lista[1])**, pides que imprima el elemento 1 de la lista. En efecto, el elemento 1 es el segundo, ya que el primer miembro de la lista es el elemento 0.

Por ello observarás que el resultado de **print(lista[1])** nos arroja “b” y no la letra “a”, esto ocurre porque Python comienza a utilizar como primer índice para sus elementos el “0”. Tenlo en consideración.



El ejemplo anterior se podría reutilizar para crear una lista de coches, animales, o una lista con elementos combinados.

John von Neumann (Budapest, Hungría, 1903-1957): Realizó contribuciones fundamentales en física cuántica, análisis funcional, teoría de conjuntos, teoría de juegos, ciencias de la computación, economía, análisis numérico, cibernética, hidrodinámica, estadística y muchos otros campos. Se le considera uno de los matemáticos más importantes del siglo XX.

En 1945, Von Neumann publica la descripción de una arquitectura de diseño para computadoras digitales, "la arquitectura von Neumann", utilizada en casi todas las computadoras modernas. Aunque vale la pena mencionar que J. Presper Eckert y John William Mauchly, también contribuyeron al concepto durante su trabajo en ENIAC.

En otras palabras, cada computadora, microcomputadora, minicomputadora y supercomputadora que conoces, es una máquina "von Neumann".

También creó el campo de los autómatas celulares sin computadores, construyendo los primeros ejemplos de autómatas autorreplicables en papel y lapiz. El concepto de constructor universal fue presentado en su trabajo póstumo "Teoría de los Autómatas Autorreproductivos.

16. Imprimir y solicitar datos

Aprendizajes esperados

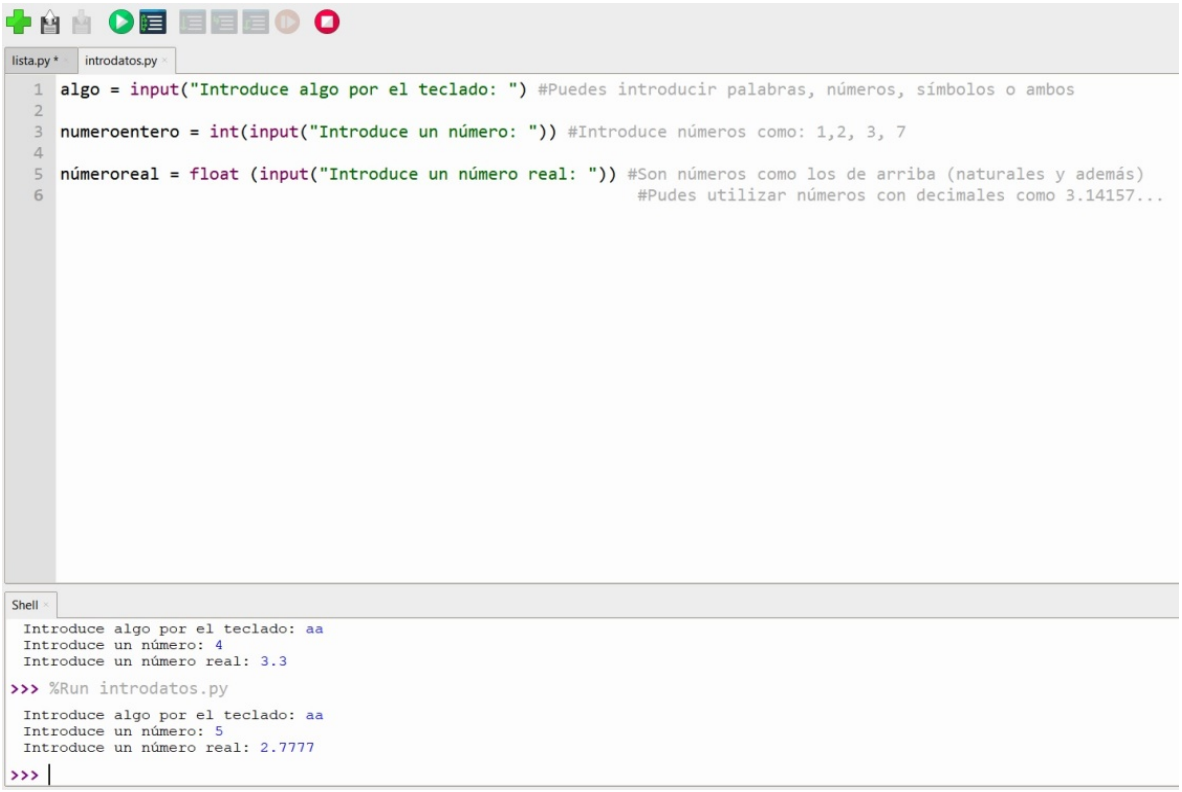
Habilidades	Medio	Contenido	Finalidad
Identifica las funciones para imprimir y solicitar datos. Reconoce los diferentes tipos de datos y las formas de nombrarlos.	Thonny Python IDE.	¿Cómo solicitar datos por teclado? ¿Cómo imprimir datos por pantalla?	Conocer las funciones para solicitar e imprimir datos. Generar un programa interactivo con diferentes tipos de datos. Reforzar sus conocimientos en el idioma inglés y en operaciones aritméticas.

16. Imprimir y solicitar datos

¿Cómo solicitar datos por teclado?

En la mayoría de los programas es necesario interactuar con el usuario para solicitarle diferentes datos. Para ello vamos a utilizar la función **input** que está diseñada para tal fin.

En sentido opuesto a la declaración de variables en la que no era necesario declarar el tipo de dato que se genera, en este caso, sí es necesario declarar lo que se quiere obtener.



Python es un lenguaje de programación interpretado y en caso de que no se obtenga el tipo de dato solicitado, se generará un error.

Prueba con el programa anterior. Introduce un número con terminación decimal en la segunda petición.

Como notarás, si se especifica un tipo flotante, la función **input** puede aceptar por igual números enteros y números reales; pues un entero es también un número real.

¿Cómo imprimir datos por pantalla?

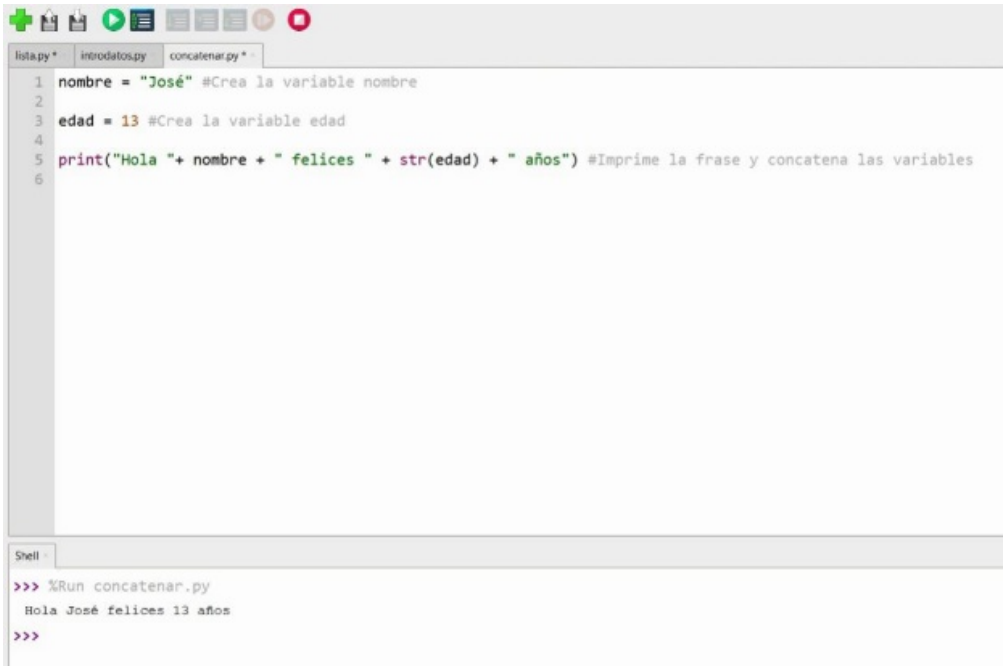
En la primera lección del curso se vio cómo imprimir el texto "Hola Mundo", y utilizaste el Shell para hacer operaciones aritméticas. Ahora haz lo mismo pero con la función **print**, como se ve a continuación.

Ahora es tiempo de que unas diferentes de datos. Por ejemplo: diseña un programa que en una única línea imprima una frase y un nombre.

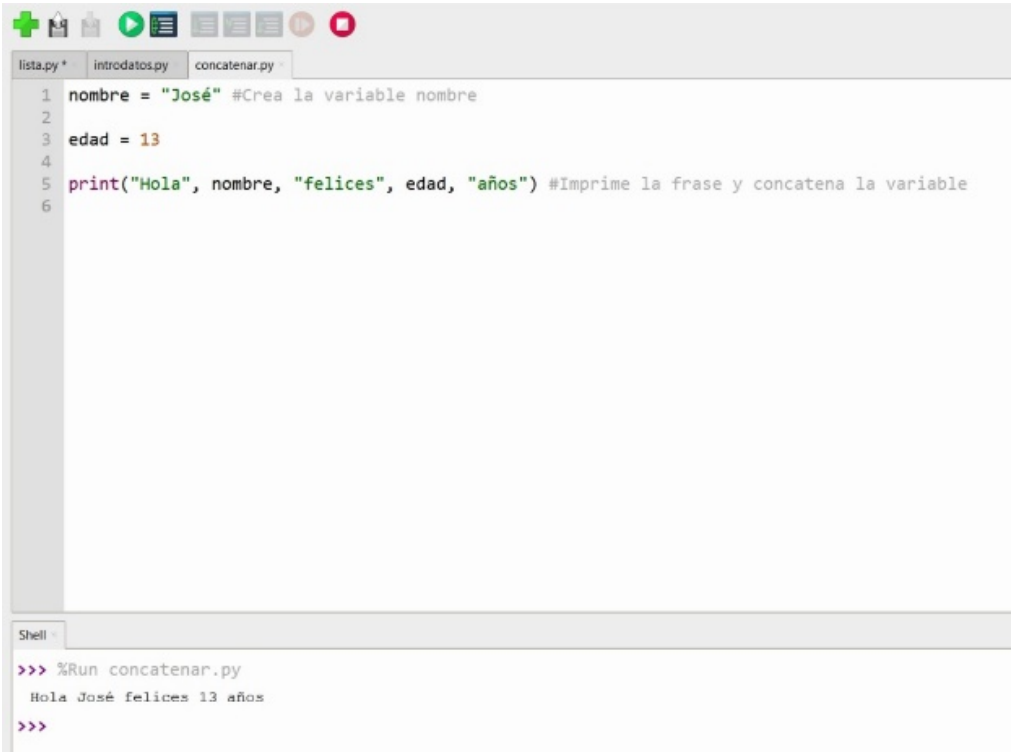
En este caso debes concatenar (unir) el valor de la frase con el valor de la variable (nombre); para esto se utiliza el símbolo reservado **"+".**

Otra forma de obtener el mismo resultado es unir los datos separándolos con comas, dentro del paréntesis. Ahora lo verás.

En caso de que quieras mostrar un texto y concatenar un número entero o real (flotante), entonces debes convertir la variable a tipo de texto y esto se logra con la función **str** (*string*-cadena).



Si utilizas la forma de comas para concatenar, no es necesario utilizar la función **str.**



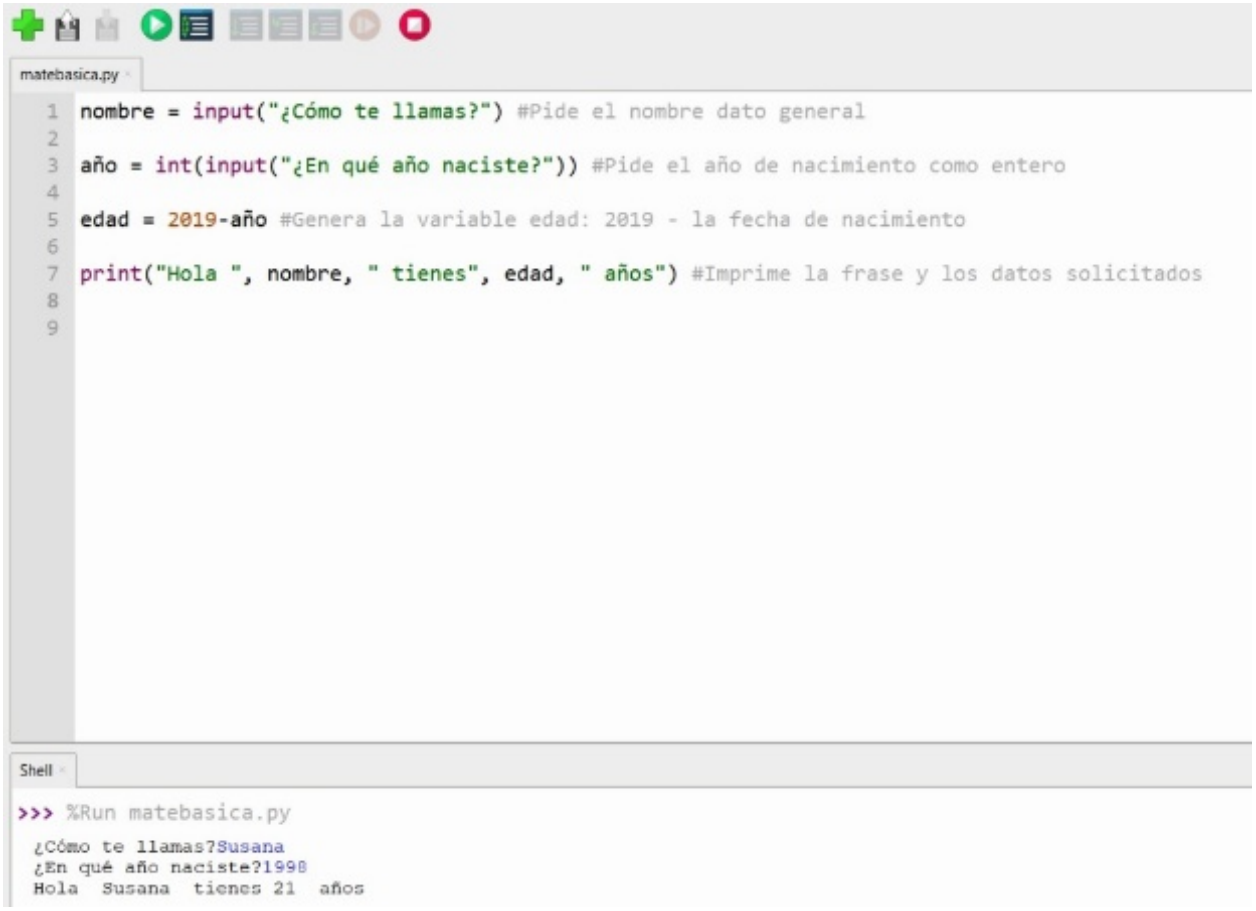
Como has visto, hay dos formas para obtener el mismo resultado. Utiliza la que más te acomode.

Reto.

Haz un programa que pida un nombre al usuario, solicite su edad, y después genere la siguiente Frase: “Hola (nombre), tienes (número) años”.

Repite el ejercicio anterior pero, ahora haz que el programa pida su fecha de nacimiento y calcule los años que tiene actualmente.

Si no te salen. No te preocupes, te dejamos el programa para que lo revises.

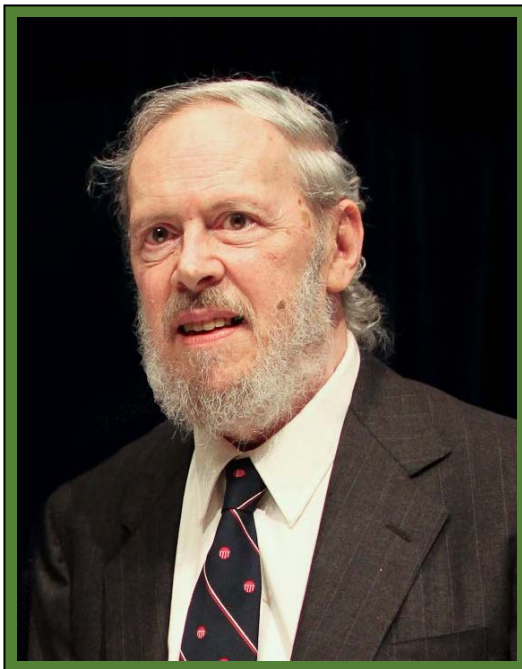


matebasica.py

```
1  nombre = input("¿Cómo te llamas?") #Pide el nombre dato general
2
3  año = int(input("¿En qué año naciste?")) #Pide el año de nacimiento como entero
4
5  edad = 2019-año #Genera la variable edad: 2019 - la fecha de nacimiento
6
7  print("Hola ", nombre, " tienes", edad, " años") #Imprime la frase y los datos solicitados
8
9
```

Shell

```
>>> %Run matebasica.py
 ¿Cómo te llamas?Susana
 ¿En qué año naciste?1998
 Hola  Susana  tienes 21  años
```



Dennis MacAlistair Ritchie (1941-2011) fue un científico de la computación estadounidense, pionero de la informática moderna, mayormente conocido por el desarrollo del lenguaje de programación “C”.

Colaboró en el diseño y desarrollo de los sistemas operativos MULTICS y UNIX, así como el desarrollo de varios lenguajes de programación como el “C”; tema sobre el cual escribió, junto a Brian Wilson Kernighan, un célebre clásico de las ciencias de la computación: “El lenguaje de programación C”.

Recibió el Premio Turing de 1983 por su desarrollo de la teoría de sistemas operativos genéricos y su implementación en la forma del sistema Unix. En 1998 le fue concedida la Medalla Nacional de Tecnología de los Estados Unidos de América.

Estos aportes convirtieron a Ritchie en un importante pionero de la informática moderna. El lenguaje C se usa ampliamente hoy día en el desarrollo de aplicaciones y sistemas operativos, y ha sido una gran influencia en otros lenguajes más modernos como el lenguaje de programación Java. Unix también ha sentado las bases de los sistemas operativos modernos, como GNU/Linux y Mac OS X, estableciendo conceptos y principios que hoy son ampliamente adoptados.

17. Condicionales

Aprendizajes esperados

Habilidades	Medio	Contenido	Finalidad
Identifica la estructura lógica para comparar datos. Reconoce las reglas para escribir un programa con condicionales.	Thonny Python IDE	¿Qué es una condición? If (Si), Else (De lo contrario), Elif (Si no se cumplió lo anterior y si…)	Conocer la estructura de una condición Conocer la sintaxis para escribir una condición Combinar lo aprendido y utilizarlo en un programa condicional.

17. Condicionales

¿Qué es una condición?

Una condición es algo que debe de ocurrir para tomar una acción. Un ejemplo en nuestra vida sería: Si mi mamá me da permiso, voy a la fiesta, si no, entonces me quedo en casa.

Si mi mamá me da permiso (condición)
Voy a la fiesta. (acción)
Si no (condición)
Entonces me quedo en casa. (acción)

En programación es lo mismo. Se tiene que escribir las condiciones necesarias para que el programa ejecute una u otra sentencia (acciones).

A diferencia que en otros lenguajes de programación, en Python no existen llaves para incluir las instrucciones de la condición. Pero como se dijo al principio cada lenguaje tiene sus particularidades y tienes que poner mucha atención a su estructura.

En Python la forma de mantener un orden y una jerarquía son las sangrías, es decir aquello que este adentro del texto anterior es lo que se debe ejecutar, De esta forma el código queda muy legible. Ve cómo quedaría el ejemplo anterior.

Si mi mamá me da permiso
Voy a la fiesta.
Si no me da permiso
Me quedo en casa.

Otra diferencia de Python sobre el resto de lenguajes de programación, es que en Python no existen los famosos ";" que se sitúan al final de las instrucciones. Sin embargo, en el caso de las condiciones, bucles o funciones debes añadir ":" para indicarle al interprete que empieza una estructura.

Otro ejemplo seria Yo tengo 5 pesos, si alguien me pregunta ¿tienes más de tres pesos? Sí, en efecto, traigo más de tres pesos. Si no traigo más de 3 pesos y alguien me pregunta ¿tienes más de tres pesos? Entonces respondo: No traigo más de 3 pesos.

Ve el ejemplo anterior descrito en 3 diferentes lenguajes (Scratch, Python, y C).

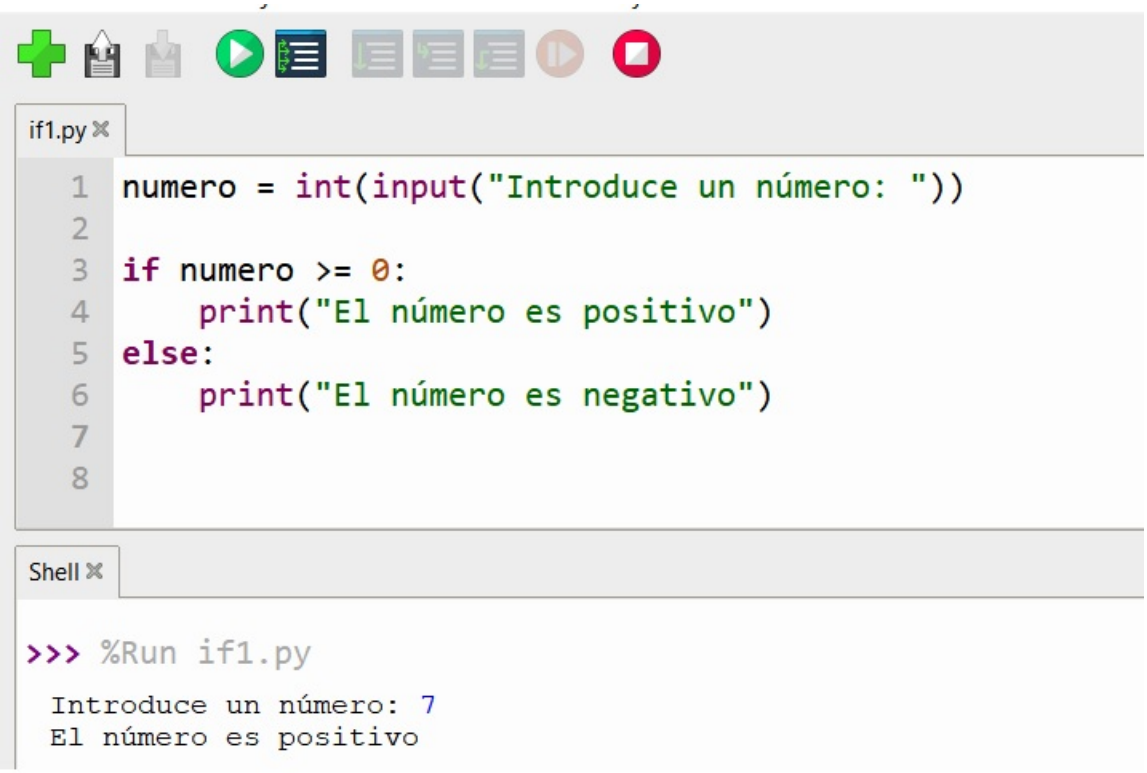

```python
numero = 5
if numero > 3:
    print("SI se cumple")
else:
    print("NO se cumple")
```

```c
var numero = 5;
if(numero > 3){
    print("SI se cumple");
}else{
    print("NO se cumple");
}
```

If (Si), Else (De lo contrario), Elif (Si no se cumplió lo anterior y si…)

Los condicionales **if, else, elif** en Python se utilizan para ejecutar una instrucción en caso de que una o más condiciones se cumplan. Un condicional es la proposición lógica cuyo valor de verdad define la manera en que realizan las acciones tu programa. Es decir, dependiendo de este valor de verdad, ocurrirá una cosa u otra, o ninguna.

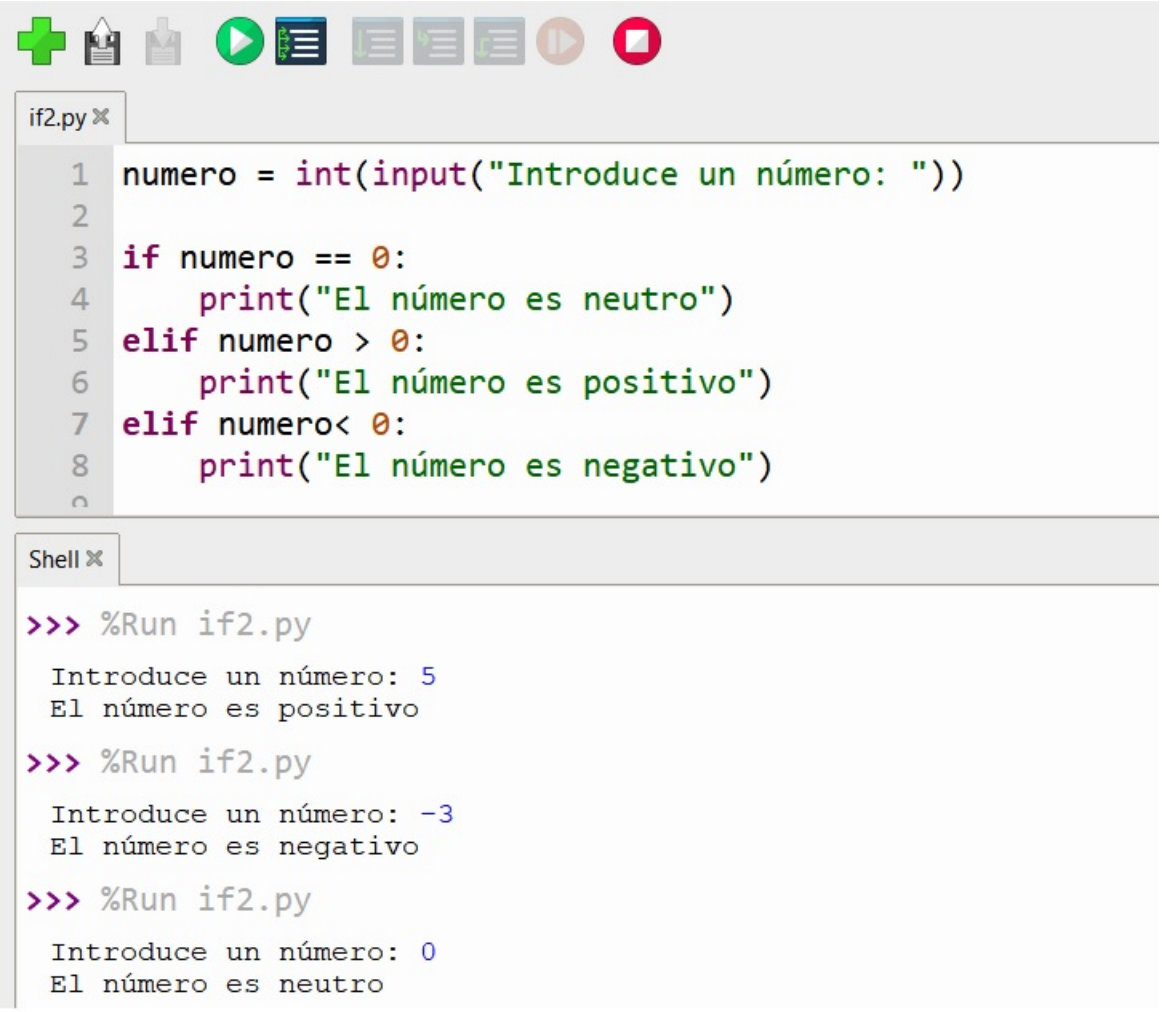
El ejemplo más sencillo lo podemos ver cuando queremos determinar si un número introducido por el usuario es positivo o negativo. La comparación en este caso consiste en comprobar si el número es mayor o igual que 0 para el caso del positivo, y en caso contrario será negativo.



Sin embargo, hay ocasiones en las que se requiere comparar más de dos opciones, por ejemplo si un número es positivo, negativo o neutro. A este tipo de condición se le conoce como condición anidada ya que aparece una condición en el interior de otra.

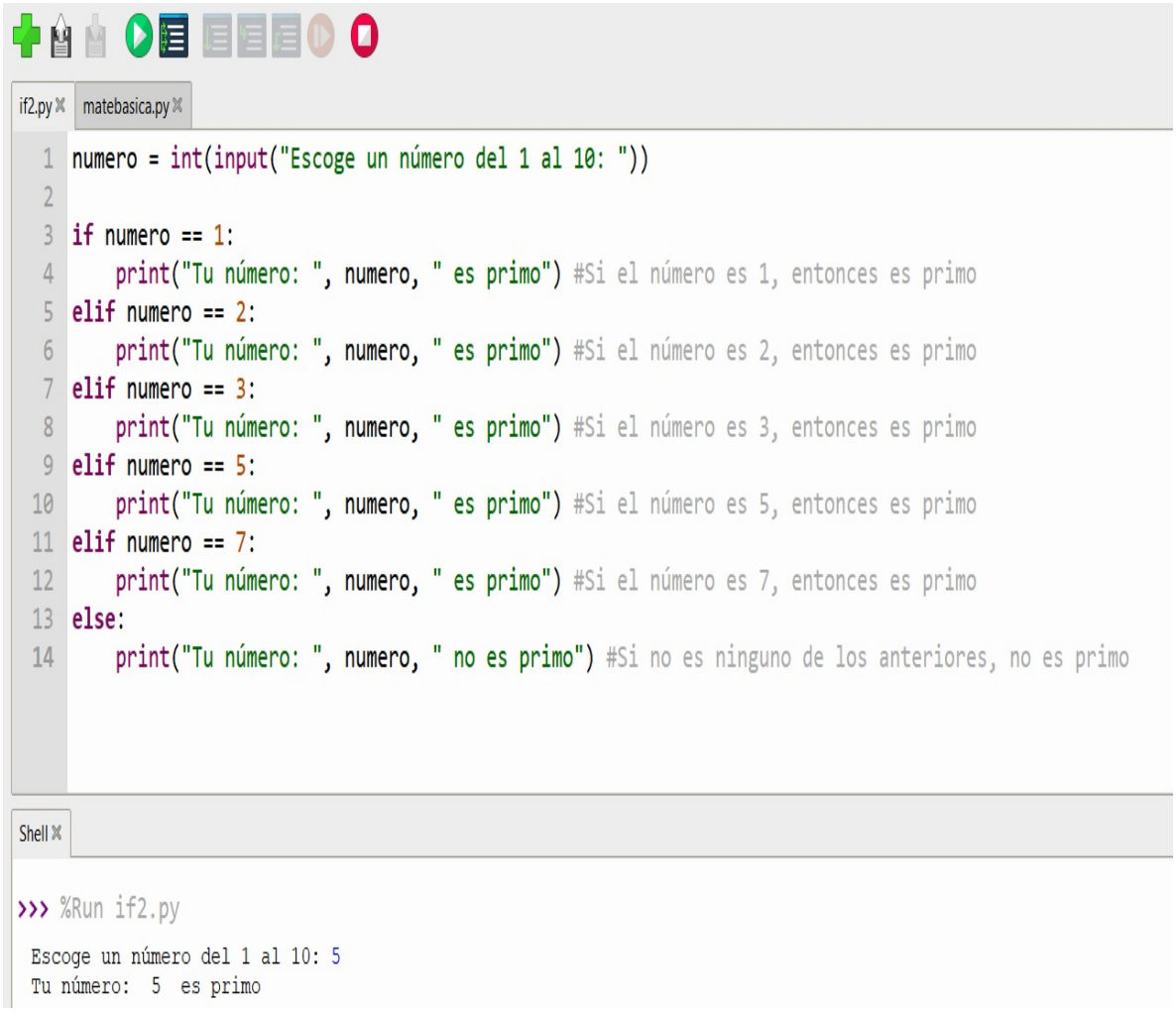
Aclaración importante: el **"=="** es un operador de **comparación** (compara si los valores son exactamente iguales). Si

colocas el **“=”** en el **if, elif o else”**, te dará un error porque este último es un operador de **asignación** (asigna un valor).



Ahora combina todo lo que has aprendido.

Genera un programa que pida un número entre el 1 y el 10 y que el programa diga si es número primo o no, recuerda que un número primo es aquel que solo es divisible entre él mismo y la unidad.



El programa anterior tiene una falla, permite que el usuario introduzca números mayores al 10 y el programa en ese caso dirá que número 11 no es un número primo, lo cual es un error, ¿verdad?

Reto:

Genera una versión del programa anterior; pero en esta ocasión, si el usuario introduce un número mayor a 10, que el programa debe mandar un mensaje al usuario indicando que el número que ha introducido no se encuentra entre el 1 y el 10.

Reto.

Genera un programa que pida un número e identifique si es par o impar.

Ayuda.

El símbolo % en Python se ocupa como operador residuo de la división entera, es decir si divido 9/3 el residuo es 0 y si divido 5/2 el residuo es 1. En python el primer ejemplo se escribiría como 9%3 == 0 y 5%2==1

¿Podrías hacer un programa que te diga si un número es primo usando este operador?

Frances Elizabeth Allen (4 de agosto de 1932) es una informática estadounidense y pionera en el campo de la "Optimización de Compiladores". Sus logros incluyen trabajo en: compiladores, optimización de código y computación paralela. También tuvo un rol importante en la creación de lenguajes de programación y códigos de seguridad para la Agencia de Seguridad Nacional de los Estados Unidos de Norteamericana.

Allen fue la primera socio de IBM y, en 2006, se convirtió en la primera mujer ganadora del Premio Turing.

17. Bucles (for, while) y funciones

Aprendizajes esperados

Habilidades	Medio	Contenido	Finalidad
Identifica qué es un ciclo. Reconoce las diferencias y usos de for y while Reconoce las reglas para escribir un programa con ciclos.	Thonny Python IDE	¿Qué es un bucle? Ciclo for Ciclo while ¿Qué es una función?	Conocer la estructura de una condición Conocer la sintaxis para escribir una condición Combinar lo aprendido y utilizarlo en un programa condicional. Reforzar el concepto de función. Crear funciones.

17. Bucles (for, while) y funciones

¿Qué es un bucle?

Un bucle o ciclo, en programación, es una secuencia que ejecuta repetidas veces un fragmento de código, hasta que la condición asignada a dicho bucle deja de cumplirse.

Los dos bucles que verás en esta práctica son: el **for** (para todo) y **while** (mientras)

Un bucle **for** es un bucle que se repite un cierto número de veces sobre los datos del programa; pueden ser números, cadenas, listas, tuplas, etc. El ciclo hará un recorrido y ejecutara una orden siempre y cuando pueda actuar sobre esos datos.

Un ejemplo sería inflar un globo con la boca, soplas hasta que esté inflado, eso puede llevarte, 3 o 5 o 10 veces, depende de tu habilidad y de tus pulmones. En este ejemplo, con un bucle **for** podrías especificar el número de soplidos para decidir parar. Ahora bien, si quisieras utilizar la condición de que tu globo este inflado para decidir parar, podrías ocupar un bucle **while.** En este caso puedes decir: mientras el globo no esté inflado, seguir soplando.

Otro ejemplo de un bucle **while**, sería un hámster dando vueltas en una rueda, ¿cuánto tiempo lo hace? Puede hacerlo hasta que se canse, o hasta que le enseñes comida, o hasta que lo asustes.

El bloque de instrucciones que se repite se suele llamar cuerpo del bucle y cada repetición se suele llamar iteración.

Iterar: significa realizar cierta acción repetidas veces. En el caso de **for** hace referencia a recorrer elementos iterables, como puede ser un diccionario en búsqueda de un elemento en particular.

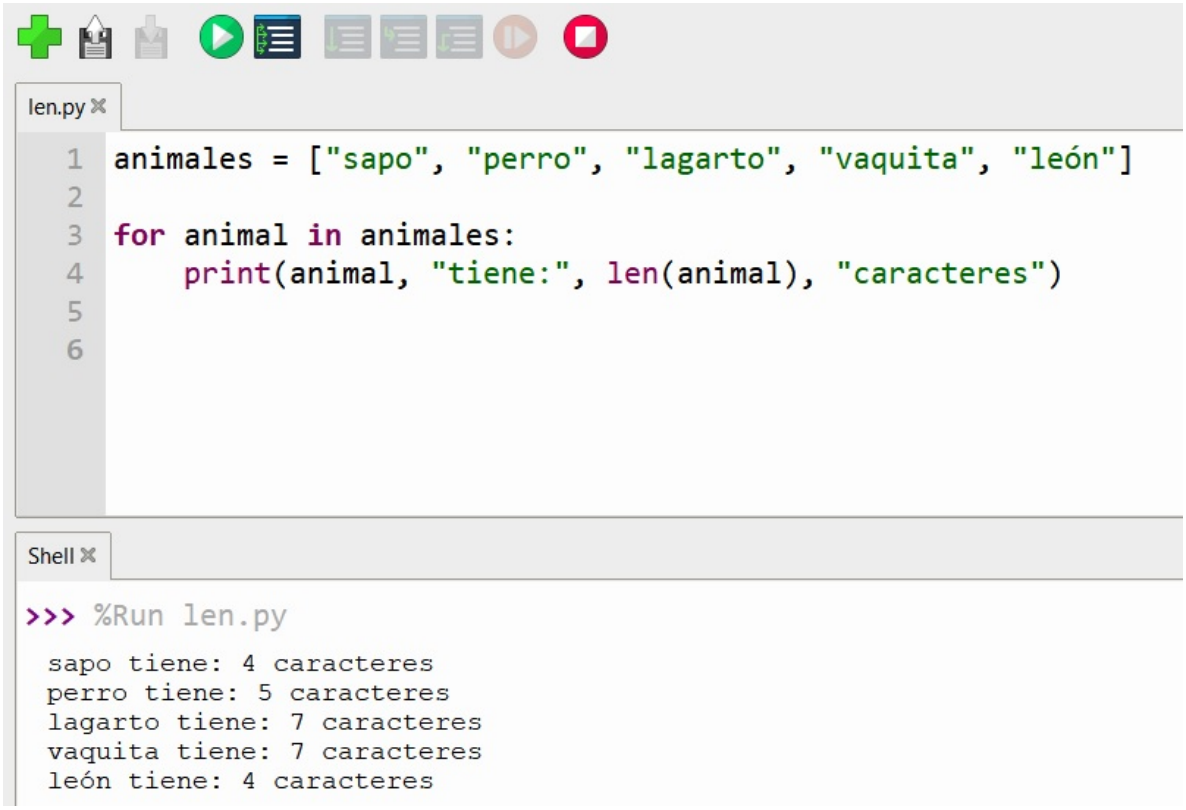
En los siguientes ejemplos puedes darte una idea de cómo usar el ciclo **for.** Ya sea definiendo un rango de valores para actuar o recorriendo una lista lista para mostrar todos los elementos ésta. Verás que es bastante intuitiva su forma de uso.

Reto:

Combina, los ejercicios anteriores. Haz una lista de números, del 1 al 16 e imprime sólo los números pares.

El siguiente ejemplo muestra cómo puedes operar con una función en una lista, en este caso queremos saber cuántos caracteres tiene cada elemento de la lista animales, eso se logra con la función len, más adelante verás muchas funciones interesantes.

..



len.py

```
1  animales = ["sapo", "perro", "lagarto", "vaquita", "león"]
2
3  for animal in animales:
4      print(animal, "tiene:", len(animal), "caracteres")
5
6
```

Shell

```
>>> %Run len.py
  sapo tiene: 4 caracteres
  perro tiene: 5 caracteres
  lagarto tiene: 7 caracteres
  vaquita tiene: 7 caracteres
  león tiene: 4 caracteres
```

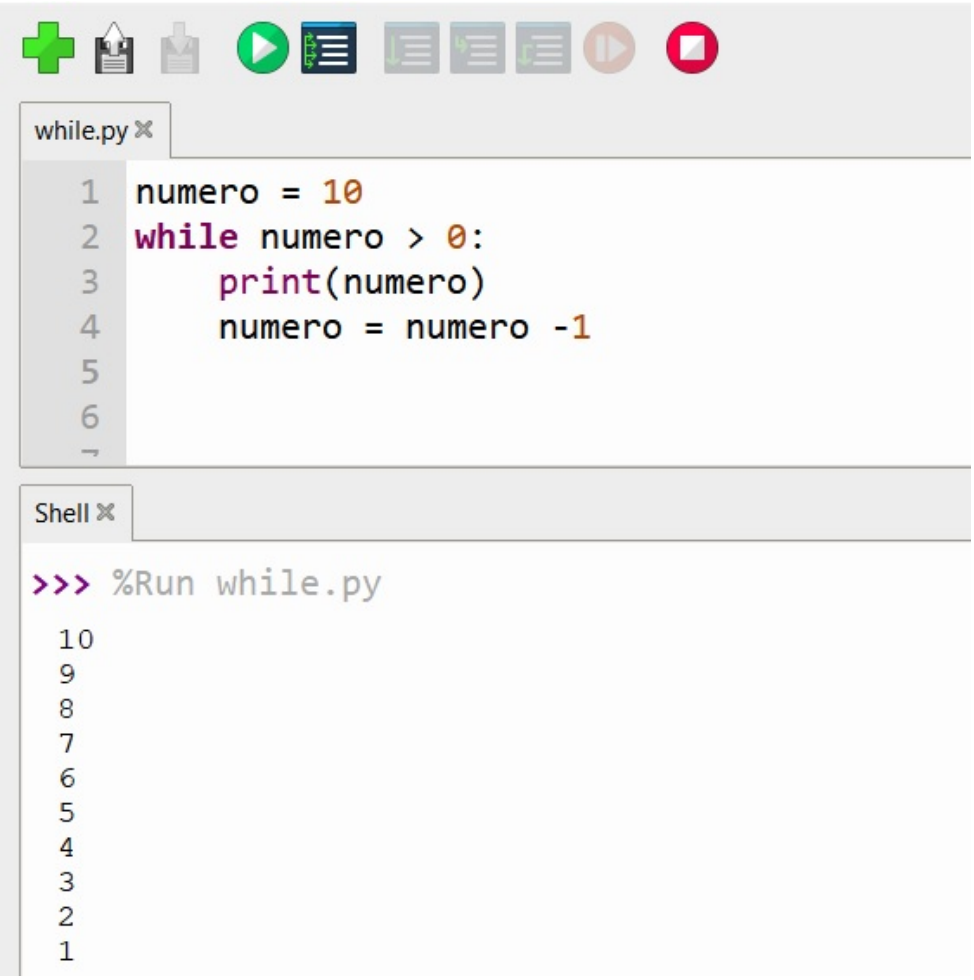
Reto:

Genera un programa que imprima la tabla de multiplicar, del 0 al 10, del número que el usuario introduzca.

Bucle (while)

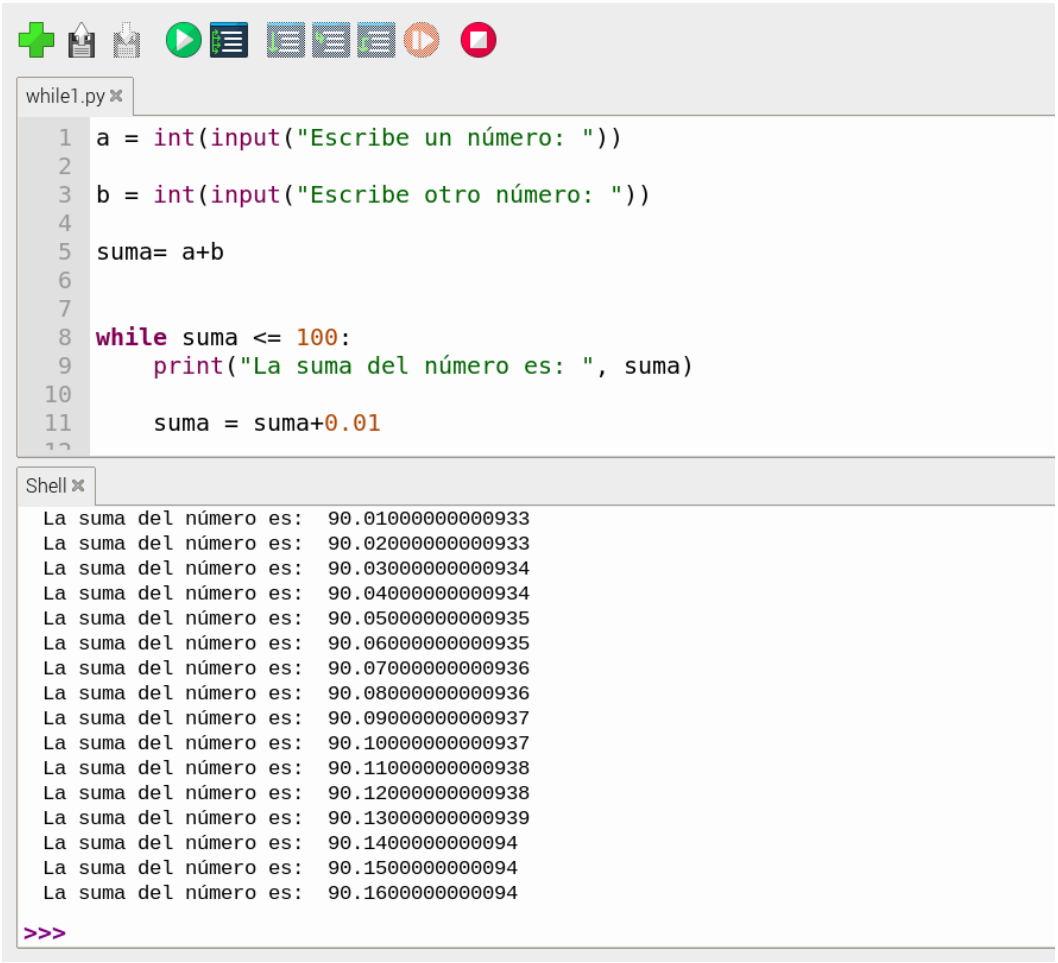
Como ya se dijo anteriormente, un bucle **while** permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (es decir, mientras la condición tenga el valor **True** o verdadero).

En este tipo de bucle hay que tener mucho cuidado, ya que es muy común caer en bucles infinitos cuando se empieza a programar. Veamos un ejemplo, fíjate como en el bucle de la siguiente imagen, se decrementa el valor de número en cada iteración.



Reto:

Pide dos números enteros y después súmalos y haz que aparezca la suma con una diferencia de 0.01 hasta que el total sea cien. En caso de que no puedas, te ponemos el ejemplo abajo.



while1.py

```
1  a = int(input("Escribe un número: "))
2
3  b = int(input("Escribe otro número: "))
4
5  suma= a+b
6
7
8  while suma <= 100:
9      print("La suma del número es: ", suma)
10
11     suma = suma+0.01
```

Shell

```
La suma del número es:  90.01000000000933
La suma del número es:  90.02000000000933
La suma del número es:  90.03000000000934
La suma del número es:  90.04000000000934
La suma del número es:  90.05000000000935
La suma del número es:  90.06000000000935
La suma del número es:  90.07000000000936
La suma del número es:  90.08000000000936
La suma del número es:  90.09000000000937
La suma del número es:  90.10000000000937
La suma del número es:  90.11000000000938
La suma del número es:  90.12000000000938
La suma del número es:  90.13000000000939
La suma del número es:  90.1400000000094
La suma del número es:  90.1500000000094
La suma del número es:  90.1600000000094

>>>
```

¿Qué es una función?

En programación una función es un conjunto aislado de instrucciones que realizan una tarea determinad, como el propio nombre indica. Python tiene incluidas muchas funciones, **print** por ejemplo es una de ellas. Lo que queremos en esta pr´actica es que aprendas a generar tus propias funciones

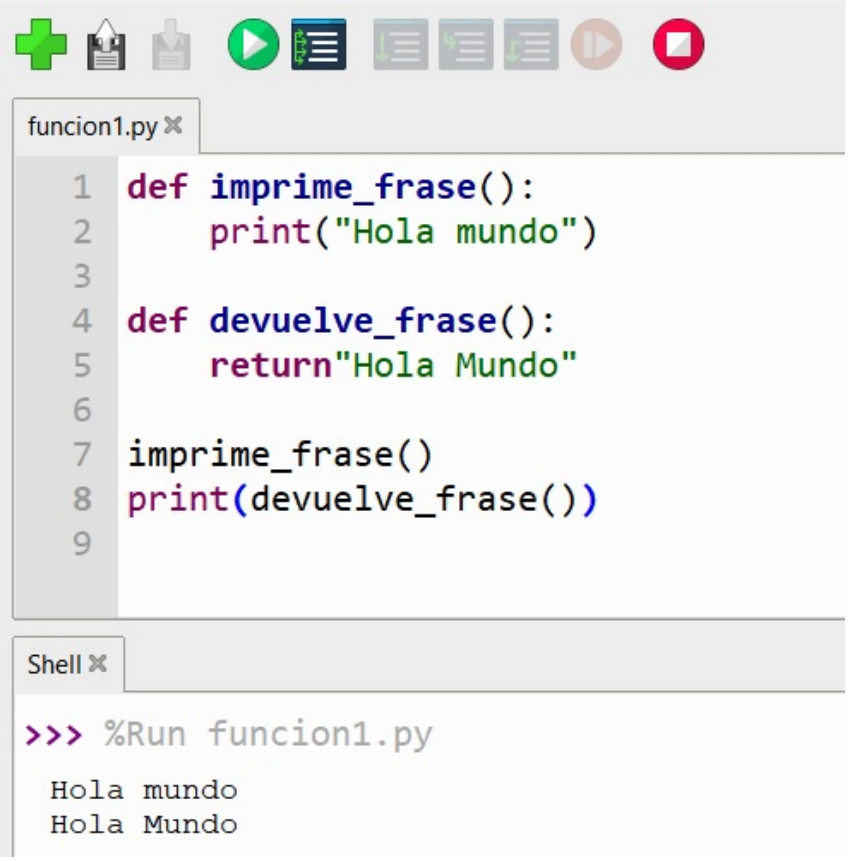
A continuación te dejamos una lista de algunas funciones útiles integradas en Python.

Función	Descripción	Ejemplo	Resultado
print()	Imprime en pantalla el argumento.	**print** («Hola Mundo»)	Hola Mundo
len()	Determina la longitud en caracteres de una cadena.	**len**(«Hola Python»)	11
range()	Crea un rango de números	x = range (5) **print** (list(x))	[0, 1, 2, 3, 4]
str()	Convierte un valor numérico a texto	**str(**22**)**	“22”
max()	Determina el máximo entre un grupo de números	x = [0, 1, 8] **print** (max(x))	8
min()	Determina el mínimo entre un grupo de números	x = [0, 1, 8] **print (**min(x))	0

Función	Descripción	Ejemplo	Resultado
sum()	Suma el total de una lista de números	**x = [0, 1, 2]** print (sum(x))	3

Una función puede recibir diferentes parámetros y puede expresarse de varias maneras.

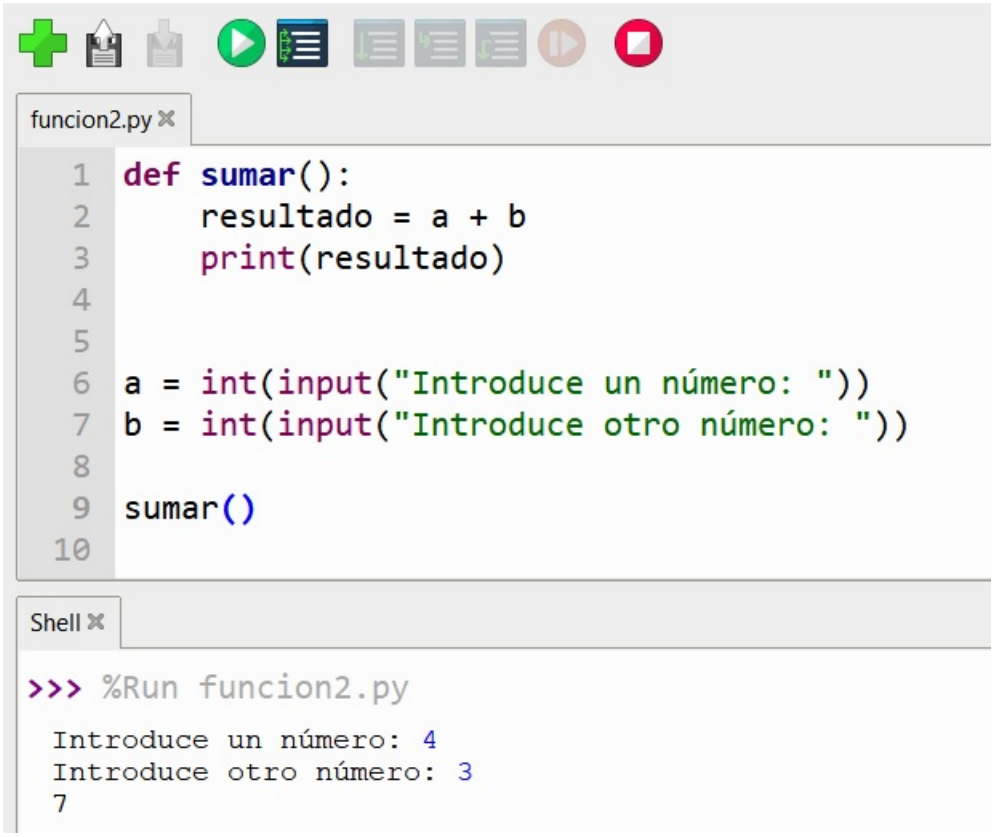
En el siguiente ejemplo se definen dos funciones que **permiten** imprimir el mensaje "Hola mundo" de maneras distintas a usar directamente la instrucción **print("Hola mundo")**, pero con el mismo resultado.

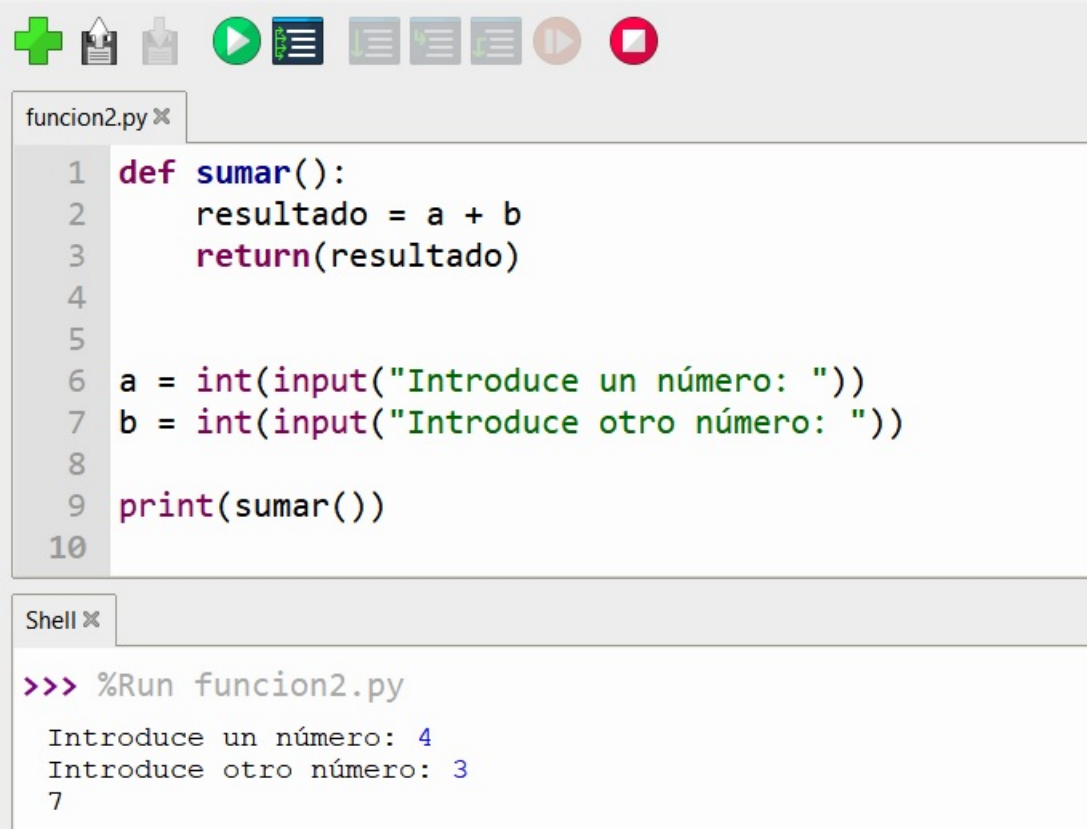


En las dos imágenes siguientes se puede observar cómo se ha creado una función llamada **sumar** la cual suma dos números y los almacena en la variable **resultado**. Por último devuelve el resultado mediante la función **print**

Cada programa es diferente: el primero imagen manda imprimir el resultad dentro de la nueva función **(sumar)** y la segunda sólo regresa el resultado, pero no lo manda imprimir, por eso es necesario introducir la función sumar en la función **print.**

.





Para revisar una gran cantidad de librerías y sus respectivas funciones, visita el siguiente link:

https://docs.python.org/3/library/

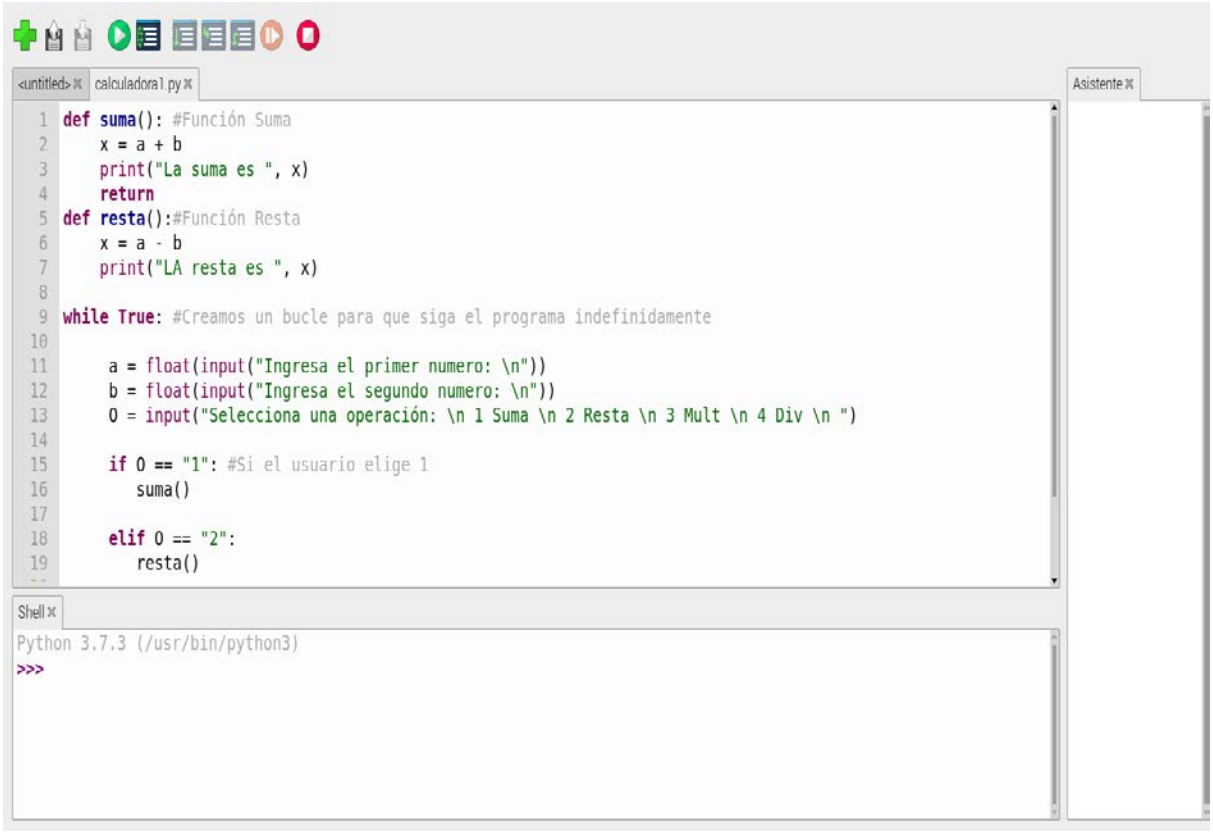
Reto.

Revisa el siguiente ejemplo y termina el ejercicio:

Obtén dos números y pide al usuario que elija una opción para realizar una de las cuatro operaciones básicas (suma, resta, multiplicación y división).

Para definir las operaciones, crea las funciones correspondientes.

.




```
def suma(): #Función Suma
    x = a + b
    print("La suma es ", x)
    return
def resta():#Función Resta
    x = a - b
    print("LA resta es ", x)

while True: #Creamos un bucle para que siga el programa indefinidamente

     a = float(input("Ingresa el primer numero: \n"))
     b = float(input("Ingresa el segundo numero: \n"))
     0 = input("Selecciona una operación: \n 1 Suma \n 2 Resta \n 3 Mult \n 4 Div \n ")

     if 0 == "1": #Si el usuario elige 1
        suma()

     elif 0 == "2":
        resta()
```

Radia Joy Perlman (1951, Portsmouth, Virginia) es una creadora de software e ingeniera de redes, experta en seguridad, más conocida como la “Madre de Internet”. Actualmente trabaja para Dell EMC en Seattle, Estados Unidos, y anteriormente estuvo trabajando para Intel, para la que consiguió más de 47 patentes.

Famosa por ser la creadora del protocolo Spanning Tree (STP), mientras trabajaba en DEC (Digital Equipment Corporation). Este protocolo es fundamental para permitir la redundancia de caminos en las redes de área local (LAN). Perlman es autora de un libro de texto sobre redes y coautora de otro sobre seguridad de redes. Tiene más de 100 patentes emitidas.

Cuando Perlman estudiaba en el MIT a finales de los 60, era una de las aproximadamente 50 mujeres estudiantes, en una clase de aproximadamente 1,000 estudiantes.

Para cuando las residencias para hombres en el MIT se convirtieron en mixtas, Perlman se mudó de las residencias para mujeres a una residencia mixta, donde se convirtió en la "mujer residente". Más tarde dijo que estaba tan acostumbrada al desequilibrio de género, que se volvió normal.

18. Turtle

Aprendizajes esperados

Habilidades	Medio	Contenido	Finalidad
Reconoce el concepto de programación orientada a objetos. Fortalece sus conceptos lógicos-espaciales.	Thonny Python IDE	¿Qué es Turtle? Comienza a dibujar	Conocer las diferentes funciones de turtle para dibujar. Se familiza con ellas. y utiliza ciclos para dibujar

18. Turtle

¿Qué es Turtle?

Python Turtle Graphics es un módulo de Python utilizado para enseñar programación gráfica mediante coordenadas.

"Turtle" es un objeto al cual se le puede dar órdenes de movimiento.

Como se explicó al principio, Python es un lenguaje multipropósito y, en este caso particular, verás un ejemplo del paradigma de Programación Orientada a Objetos (POO). Con este paradigma, los objetos son entidades con un determinado estado o comportamiento, y pueden ser manipulados en tiempo de ejecución (como lo hiciste con Scratch).

Las funciones principales para animar el objeto "tortuga" son las siguientes:

forward(distance): Avanza una determinada cantidad de píxeles.
backward(distance): RetrocedE una determinada cantidad de píxeles.
left(angle): Gira hacia la izquierda un determinado ángulo.
right(angle): Gira hacia la derecha un determinado ángulo.

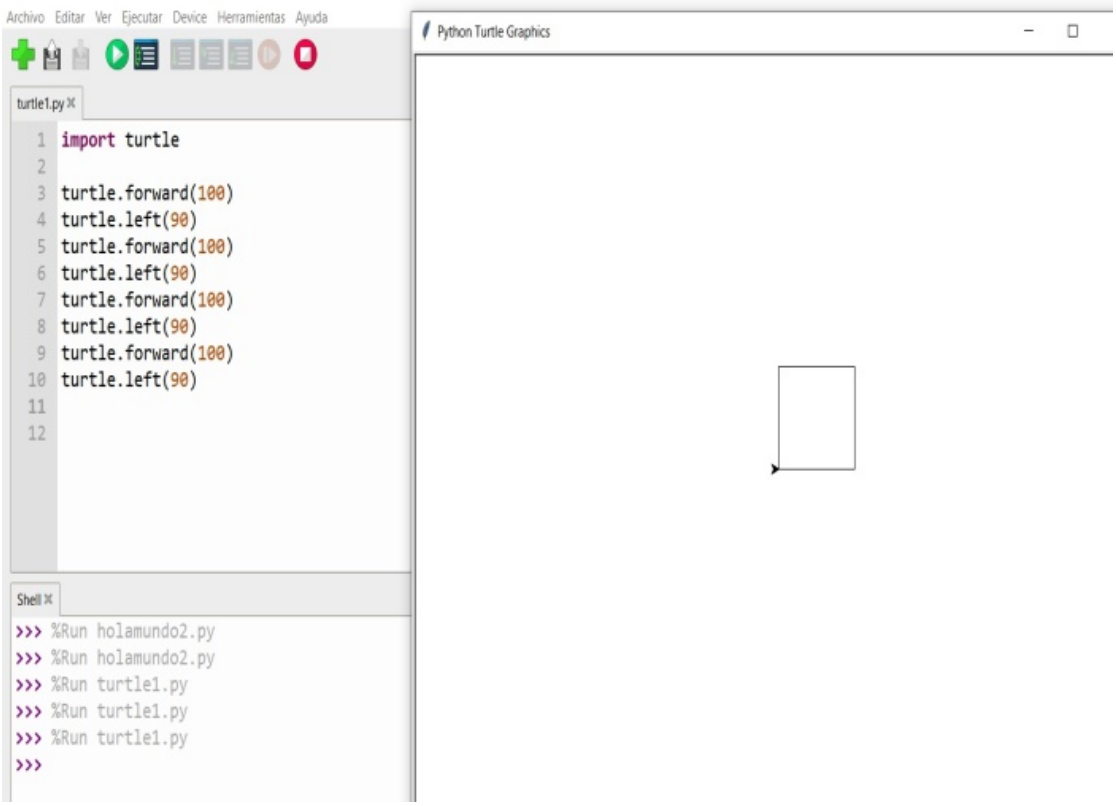
Para desplazarte de un punto a otro, ya sea dejando o sin dejar rastro (es decir dibujar o no por donde se está moviendo), debes utilizar las siguientes funciones:

home(distance): Desplaza al origen de coordenadas.
goto((x, y)): Desplaza a una coordenada en concreto.
pendown():Baja el lápiz para mostrar el rastro (dibujar).
penup():Sube el lápiz para no mostrar el rastro (no dibujar).

Por último, para cambiar la forma de tu puntero de dibujo o el color o tamaño del lápiz:

shape('turtle'): Cambia al objeto tortuga.
pencolor(color): Cambia al color especificado.
pensize(dimension): Tamaño de la punta del lápiz.

En la siguiente imagen verás un programa que dibuja una figura geométrica simple: cuadrado

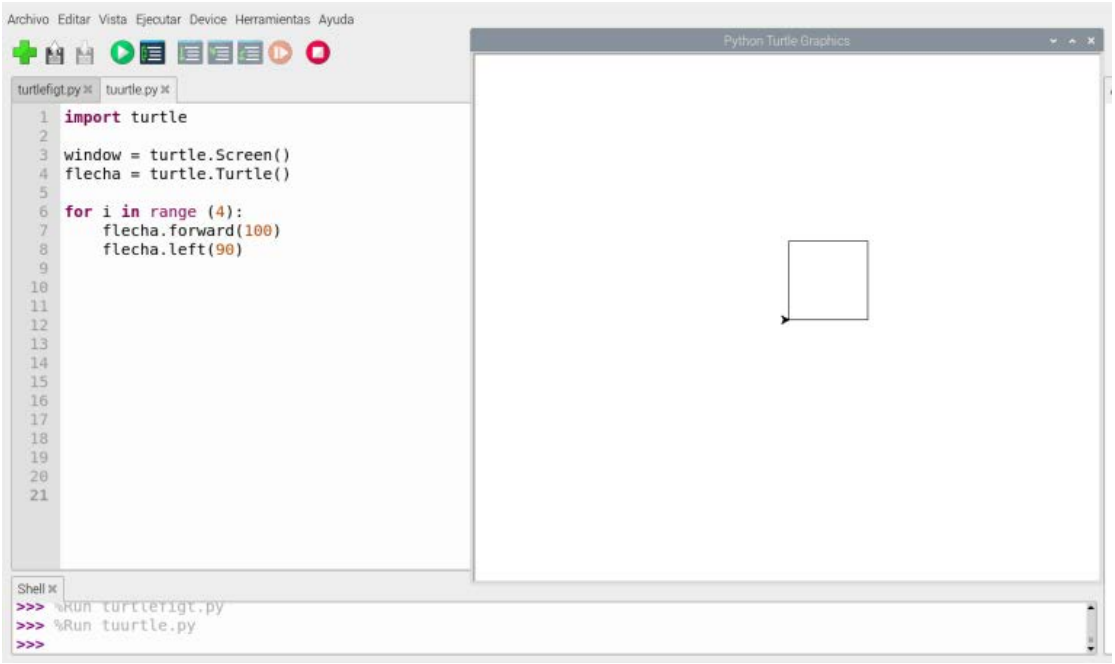


Lo primero que se hizo fue importar la librería **turtle**. Después utilizaste las funciones: turtle.foward y turtle.left para avanzar 100 pasos y girar a la izquierda 90 grados respectivamente; todo se repitió cuatro veces.

.

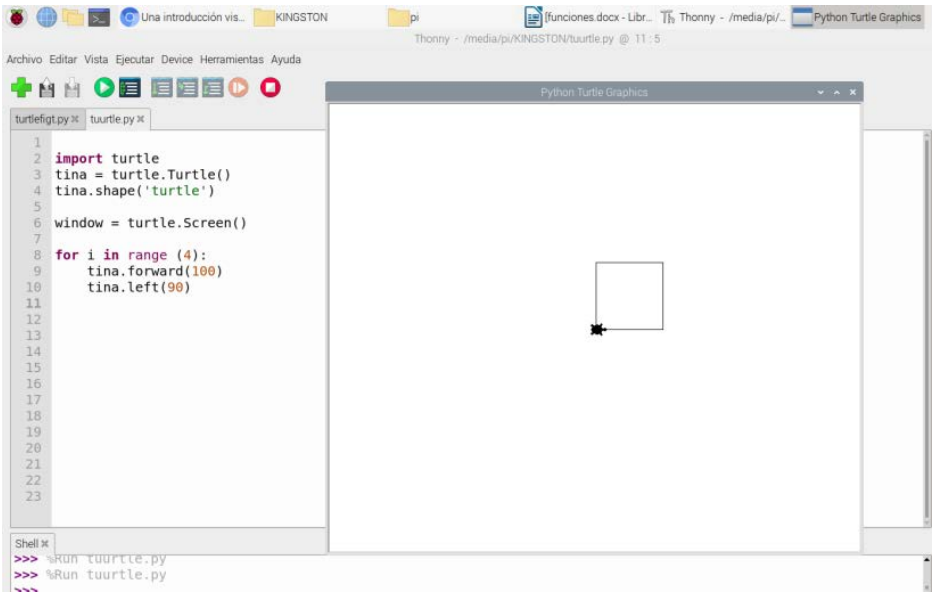
Como observaras, como no se pidió subir la pluma, se dibujó todo el trayecto y, al final, nuestro puntero quedó de nuevo en el punto de inicio.

Puedes ahorrarte muchos pasos si utilizas un ciclo **for**. Puedes definir tus variables asignándoles el funcionamiento de **turtle**. Por ejemplo, define como **turtle** la variable flecha para guiar tu dibujo.

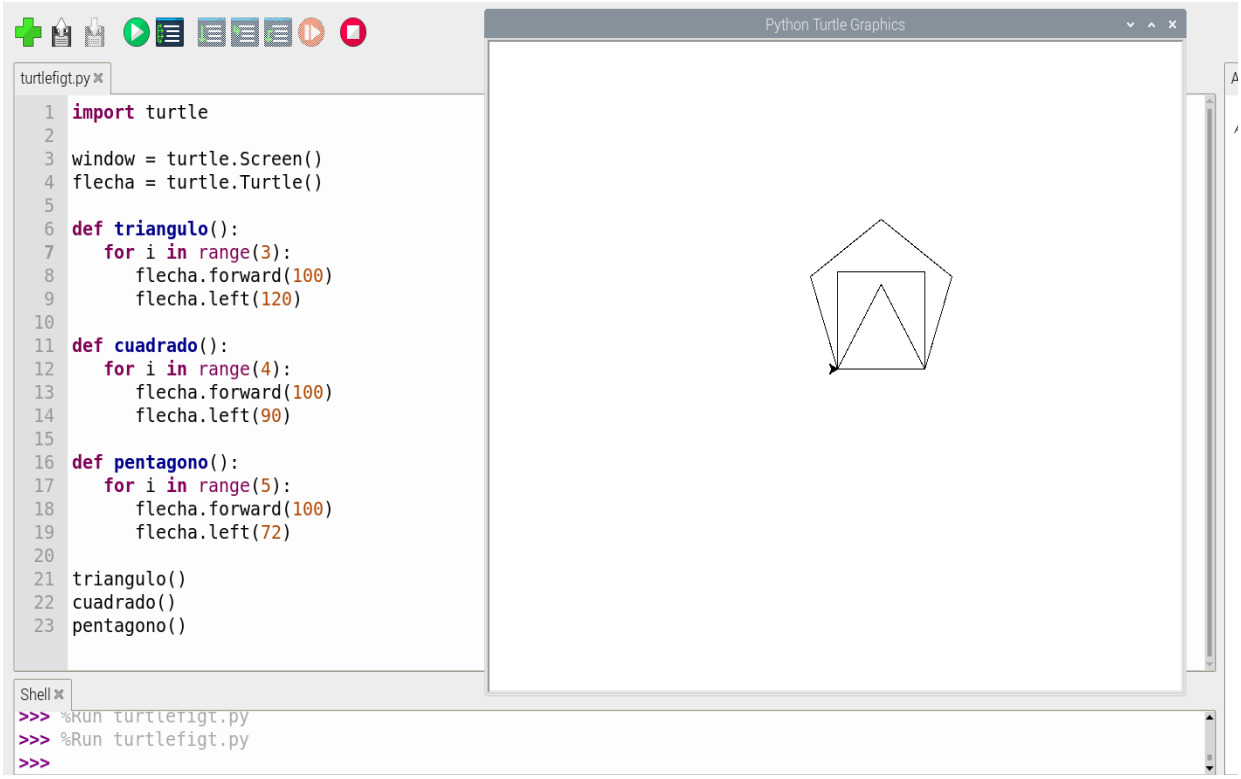


Ahora cambia la forma de triángulo de tu puntero por la de tortuga. En esta ocasión define **tina** como una instancia de turtle para trabajar con ella y cambiar su forma con **tina.shape**

.



Para el siguiente ejercicio, Genera tres figuras que se encuentren una dentro de la otra: un triángulo, un cuadrado y un pentágono. Hay varias formas de hacerlo, pero si utilizas lo que hemos visto hasta ahora, podrás ahorrarte mucho tiempo.



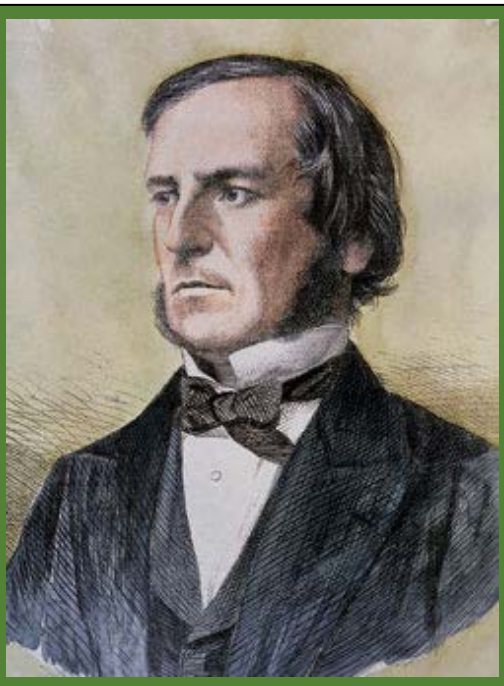
En el programa anterior, importaste librerías, definiste variables, creaste funciones con ciclos relacionados al número de lados de cada figura y después las llamaste para controlar tu dibujo. ¡Parece que estás listo para imaginarte programas complejos e interesantes!

George Boole (Lincoln, Lincolnshire, Inglaterra, 1815-1864) fue un matemático y lógico británico.

Inventor del álgebra de Boole, la cual sienta los fundamentos de la aritmética computacional moderna.

Boole es considerado como uno de los fundadores del campo de las Ciencias de la Computación. En 1854 publicó “An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities”, donde desarrolló un sistema de reglas que le permitían expresar, manipular y simplificar problemas lógicos y filosóficos, cuyos argumentos admiten dos estados (verdadero o falso), por procedimientos matemáticos.

Se podría decir que es el padre de los operadores lógicos simbólicos y que gracias a su álgebra, hoy en día es posible operar simbólicamente para realizar operaciones lógicas.



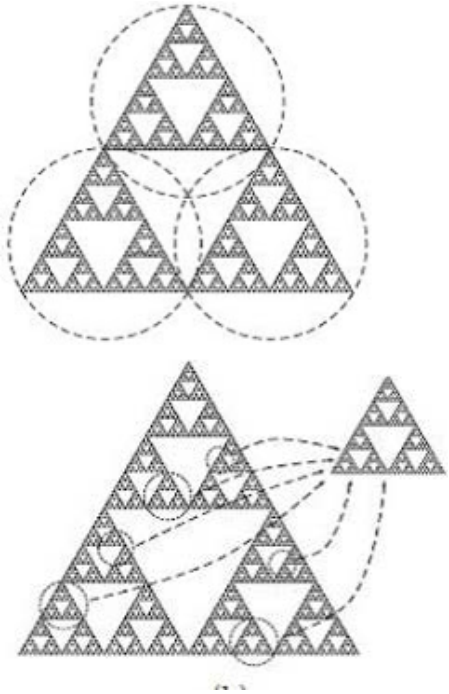
.

19. Fractales

Aprendizajes esperados

Habilidades	Medio	Contenido	Finalidad
Reconoce el concepto de iteración, Visualiza un fractal como un objeto auto semejante a escala. Deduce los parámetros entre los cuales el objeto representa una hoja.	Thonny Python IDE	¿Qué es un fractal? ¿Cómo crear un fractal? Parámetros. Características	Conocer las diferentes expresiones matemáticas para generar un fractal, así como las características y semejanzas que existen en la naturaleza y en su geometría.

19. Fractales



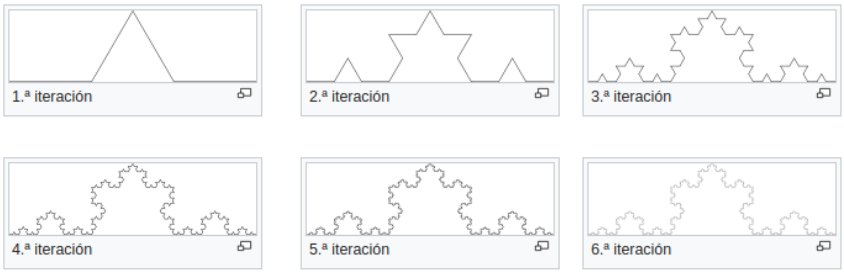
Un fractal es un objeto geométrico cuya estructura se repite como copia de si mismo a diferentes escalas.

El término fue propuesto por el matemático Benoît Mandelbrot en 1975 y deriva del latín *fractus*, que significa quebrado o fracturado. Muchas estructuras naturales son de tipo fractal. La propiedad matemática clave de un objeto genuinamente fractal es que su dimensión métrica fractal es un número racional no entero. Lo último sonó muy raro y misterioso ¿verdad?, trata de investigar un poco más sobre esos temas.

¿Cómo crear un fractal?

Un fractal muy conocido e intuitivo es la curva de Koch.

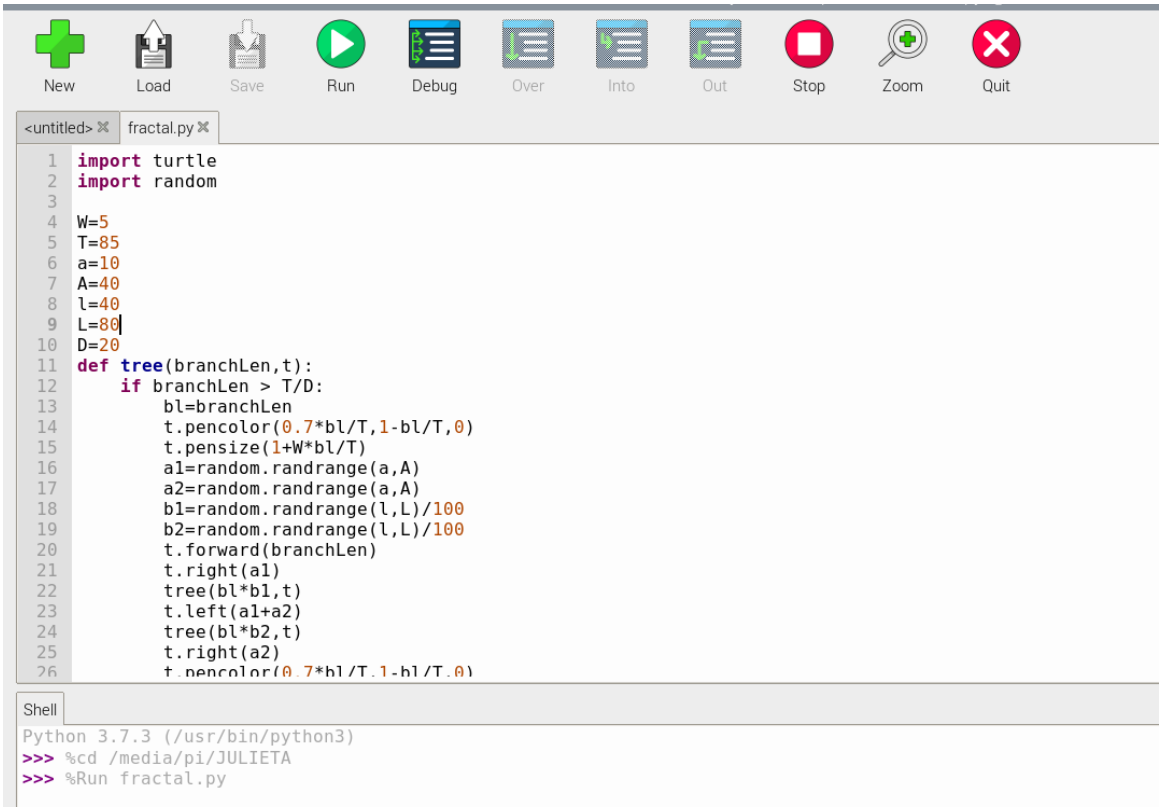
Se toma un segmento, se le divide en tres partes iguales, se remplaza la parte central por dos partes de igual longitud haciendo un ángulo de 60 grados. Luego, con los cuatro segmentos nuevos, se procede de la misma manera, lo que da lugar a 16 segmentos más pequeños en la segunda iteración. Y así sucesivamente. La siguiente figura representa las seis primeras etapas de esta construcción. La última curva es una buena aproximación de la curva final.

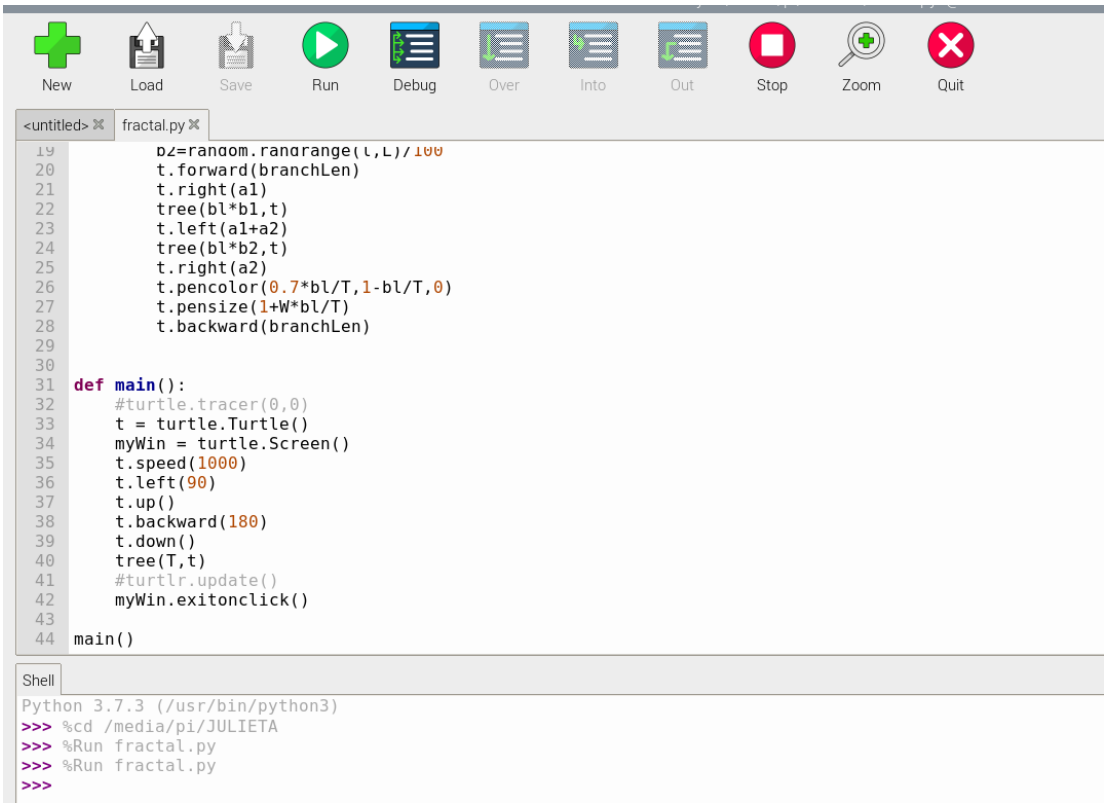


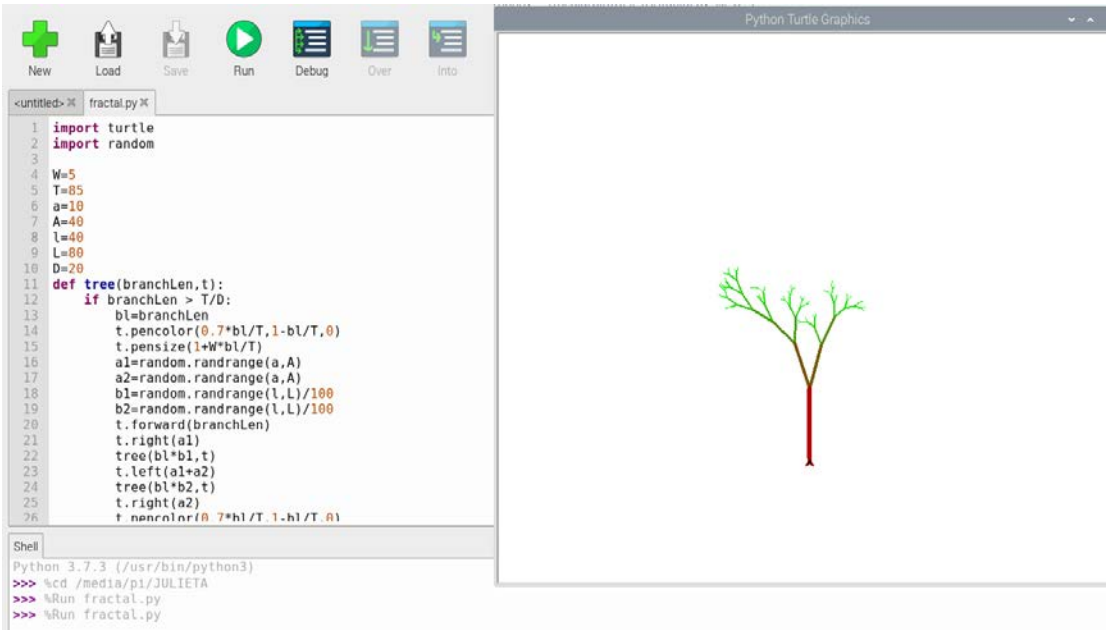
Tu primer fractal.

Si observas la construcción de la curva de Koch descubrirás que en cada paso de la construcción, cuando se crea un segmento nuevo, tendrás que hacer el mismo proceso para cada segmento creado. Eso lo puedes hacer de manera iterativa o, usando una función que se llame a sí misma en una nueva escala de tamaño hasta que considere que no deba seguir haciéndolo. Este tipo de funciones se llaman recursivas. Está fuera del alcance de esta práctica explicar a detalle el concepto de recursividad, pero bastará decir que las funciones recursivas se llevan muy bien con los fractales.

Ahora veras un ejemplo de recursión para construir un fractal muy sencillo e interesante. Abre Thonny Python y escribe el siguiente código. Léelo cuidadosamente y comenta con tus compañeros y tu facilitador lo que observas de su comportamiento.




New
Load
Save
Run
Debug
Over
Into
Out
Stop
Zoom
Quit
<untitled>
fractal.py
b2=random.randrange(l,L)/100
t.forward(branchLen)
t.right(a1)
tree(bl*b1,t)
t.left(a1+a2)
tree(bl*b2,t)
t.right(a2)
t.pencolor(0.7*bl/T,1-bl/T,0)
t.pensize(1+W*bl/T)
t.backward(branchLen)
def main():
#turtle.tracer(0,0)
t = turtle.Turtle()
myWin = turtle.Screen()
t.speed(1000)
t.left(90)
t.up()
t.backward(180)
t.down()
tree(T,t)
#turtlr.update()
myWin.exitonclick()
main()
Shell
Python 3.7.3 (/usr/bin/python3)
>>> %cd /media/pi/JULIETA
>>> %Run fractal.py
>>> %Run fractal.py
>>>


New
Load
Save
Run
Debug
Over
Into
<untitled>
fractal.py
import turtle
import random
W=5
T=85
a=10
A=40
l=40
L=80
D=20
def tree(branchLen,t):
if branchLen > T/D:
bl=branchLen
t.pencolor(0.7*bl/T,1-bl/T,0)
t.pensize(1+W*bl/T)
a1=random.randrange(a,A)
a2=random.randrange(a,A)
b1=random.randrange(l,L)/100
b2=random.randrange(l,L)/100
t.forward(branchLen)
t.right(a1)
tree(bl*b1,t)
t.left(a1+a2)
tree(bl*b2,t)
t.right(a2)
Shell
Python 3.7.3 (/usr/bin/python3)
>>> %cd /media/pi/JULIETA
>>> %Run fractal.py
>>> %Run fractal.py
Python Turtle Graphics

Benoît Mandelbrot: (Varsovia, Polonia, 1924- 2010): Matemático polaco naturalizado francés y estadounidense, conocido por sus trabajos sobre los fractales. Es considerado como el principal responsable del auge de este campo de las matemáticas en la década de los setenta. Y reconocido por haber presentado uno de los dos ejemplos más famosos de la geometría fractal: el "conjunto de Mandelbrot" y los "conjuntos de Julia". Estos últimos fueron presentados por Gaston Julia.

En 1967, Mandelbrot publicó el artículo "¿Cuánto mide la costa de Gran Bretaña?" en la revista Science, donde se exponen sus primeras ideas sobre los fractales; y en 1982 publicó su libro "Fractal Geometry of Nature", en el que explicaba sus investigaciones en este campo.

El profesor Mandelbrot se interesó por cuestiones que poco habían preocupado a los científicos en esa época: los patrones por los que se rigen las formaciones naturales, como fracturas grietas, litorales, copos de nieve, etc. En muchos aspectos, los fractales modelan estos comportamientos naturales y suelen ser asimilados por los sentidos del ser humano, con mayor naturalidad que los objetos basados en la geometría euclidiana.

GOBIERNO DE LA CIUDAD DE MÉXICO

SECRETARÍA DE EDUCACIÓN, CIENCIA, TECNOLOGÍA E INNOVACIÓN

Raspberry Pi

AEFCM
AUTORIDAD EDUCATIVA FEDERAL EN LA CIUDAD DE MÉXICO

Raspbian

python™

scratch

Este material forma parte de una iniciativa que pretende generar en niños y jóvenes, de una manera lúdica, el gusto por la programación y el desarrollo de tecnología con base en herramientas abiertas y de bajo costo. Estamos convencidos de que esta estrategia les brindará, como agradable efecto secundario, una estructura de pensamiento lógico que les preparará para desarrollarse en el campo de las ciencias, la matemática y la ingeniería.

Algunas de las prácticas de este documento fueron inspiradas en la red de clubes de código de la "Fundación Raspberry Pi" y en la iniciativa "Programo Ergo Sum".

https://projects.raspberrypi.org/en

https://www.programoergosum.es