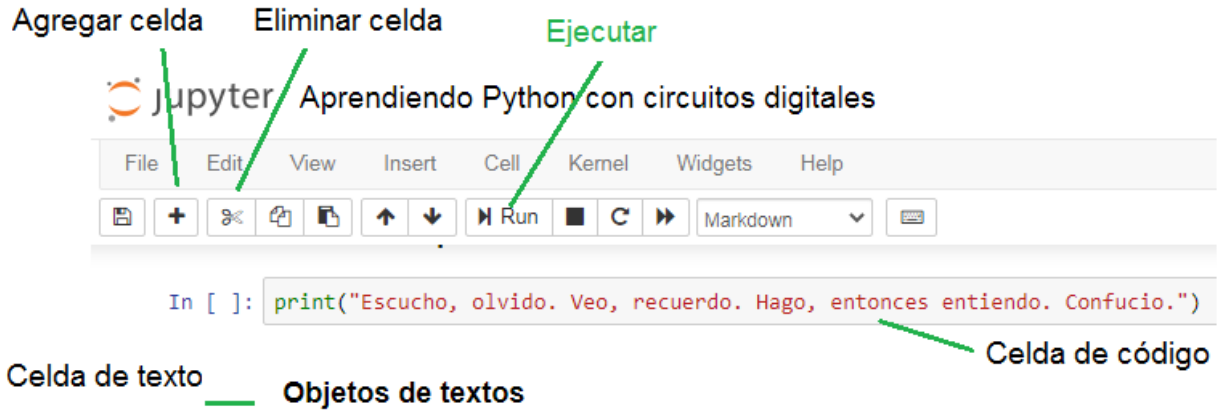
Aprendiendo Python con circuitos digitales

Lectura previa para el proyecto

Python es en lenguaje de programación y **Jupyter Notebook** es un entorno de edición. En Jupyter Notebook los programas de Python se editan y ejecutan en celdas. Cada celda funciona como una unidad de programación. Las celdas se organizan secuencialmente pero pueden ser ejecutadas en cualquier orden. Al escribir códigos en las celdas los caracteres son coloreados automáticamente para distinguir elementos del lenguaje y el editor completa caracteres como paréntesis y comillas. Para ejecutar el código de una celda se pulsa el botón **Run** de la barra de herramientas o se pulsan simultáneamente las teclas Mayúsculas y Entrar (Windows) o Control + Entrar (Mac). Estaremos trabajando con la versión 3.7 de Python en Jupyter Notebook.



En Python podemos ejecutar **instrucciones** con varios **tipos** de datos, **sentencias**, **funciones**, **módulos**, etc. Las funciones realizan tareas comunes y específicas. Se definen con **parámetros** y se utilizan con **argumentos**, que pueden ser valores de **objetos**, **expresiones** o resultados de otras funciones. Los objetos almacenan datos de algún tipo y se operan con otros objetos de acuerdo a las reglas del tipo o clase a la que pertenecen. Los tipos comunes son **str** para textos (cadenas de caracteres), **int** para números enteros, **float** para números decimales, **bool** para valores lógicos, **list** para listas, **range** para rangos, entre otros.

Los textos se escriben entre comillas simples o dobles, los números decimales llevan punto, los números enteros no tienen punto decimal. Los valores lógicos son **True**, verdadero, y **False**, falso.

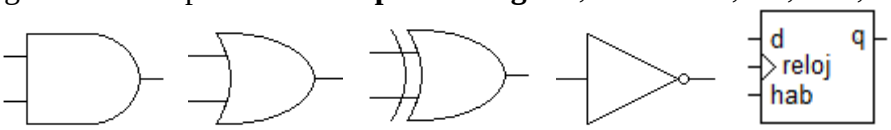
Las **clases** permiten crear objetos con sus propios **datos** e igual **comportamiento**. Las funciones definidas en clases se llaman métodos, que luego son ejecutados por los objetos de las clases. La **programación orientada a objetos** consiste en modelar clases de objetos creando métodos y definiendo sus datos, y procesar información interconectando los objetos de varias clases mediante sus métodos. Las clases pueden heredar propiedades de otras clases para especializar, eliminar o aumentar los atributos, datos y métodos. Por ejemplo, una clase Estudiante puede derivar de una clase Ciudadano, y otra clase Profesor también derivar de Ciudadano, y cada clase derivada maneja los datos y métodos provistos por Ciudadano, como nombre, apellido, edad, etc. junto con los métodos para administrarlos y procesarlos. Por ejemplo, Estudiante puede agregar cursos y notas, y Profesor puede agregar sueldos y horario de trabajo.

Un **algoritmo** describe los pasos para conseguir una solución o las tareas que realiza una función, método o programa.

Un **cronograma** muestra señales en el tiempo. En nuestro caso mostraremos señales digitales utilizando funciones de un módulo gráfico, matplotlib.pyplot.

Los **circuitos digitales** realizan **operaciones lógicas** con datos binarios (0, 1) que representan números en base dos o procesos con dos estados. Los **bits** son **dígitos binarios**, y los **números binarios** son grupos de bits. Los bits más significativos o de mayor peso suelen estar al principio y los menos significativos al final de un número. En Python las listas de números se indizan desde cero. Este aspecto es importante recordar cuando modelemos números binarios con listas.
Los circuitos digitales reciben **entradas**, las procesan y producen **salidas**. Las **operaciones aritméticas** como sumas o multiplicaciones son realizadas combinando operaciones lógicas. Estas propiedades permiten modelar circuitos utilizando los operadores lógicos de Python y encapsular las operaciones definiendo clases como modelos de los circuitos.

Los circuitos lógicos más simples se llaman **puertas lógicas**, como **And**, **Or**, **Xor**, **Not** y **Registro**.



El comportamiento de estos objetos los modelaremos mediante cuatro pasos: **inicialización**, **ingreso**, **procesamiento** y **salida** de datos. Es un método general que puedes aplicar con otros modelos de objetos y procesos, por ejemplo en un sistema de planillas, un juego de vídeo, proceso industrial, clasificación de imágenes, etc.

A partir de los circuitos básicos vamos a modelar circuitos más complejos, como un **sumador** completo, un sumador de números de tres bits, **registro** de tres bits, **contador** de tres bits y un **semáforo**. Los diagramas esquemáticos de estos circuitos aparecen en la última página. Sugiero que los imprimas o los tengas al alcance para realizar las tareas del proyecto.

Usaremos dos modelos para los circuitos más complejos. El **modelo estructural** describe el comportamiento del circuito siguiendo un diagrama esquemático. El **modelo funcional** describe al circuito algorítmicamente, solo teniendo en cuenta la relación directa entre las salidas con las entradas. Este mecanismo permite verificar el funcionamiento del circuito comparando las salidas de ambos modelos cuando reciben las mismas entradas. También es posible crear más modelos, por ejemplo uno que tenga retardos, otro que modele fallas aleatoriamente, uno que sea afectado por el tiempo de funcionamiento, etc. Esta metodología puede aplicarse para objetos de otros contextos, por ejemplo, para modelar procesos con diferentes grados de complejidad de variables y cálculos matemáticos, la cantidad de datos, el método numérico, etc.

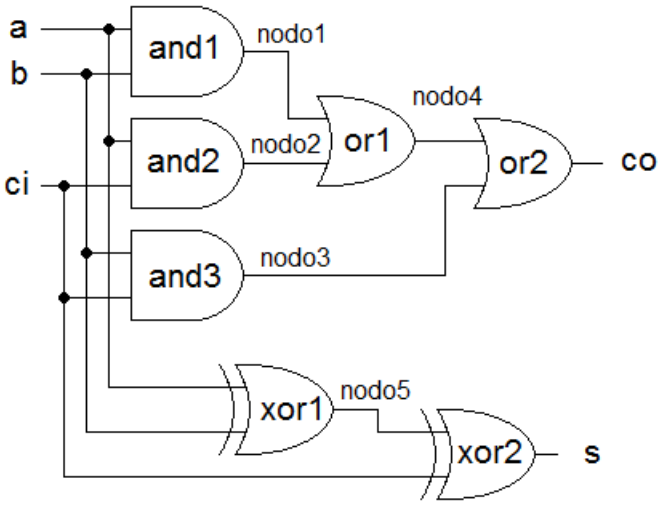
Durante el proyecto es muy importante que realices por propia cuenta los ejemplos de programación en nuevas celdas. Pausa los vídeos frecuentemente para revisar los comentarios y reproducir los códigos, y experimenta cambiándolos o creando nuevos programas. Por el corto tiempo del proyecto no es posible mencionar todos los detalles. Puedes consultar las referencias siguientes, y por supuesto buscar más información en Internet.

Recursos

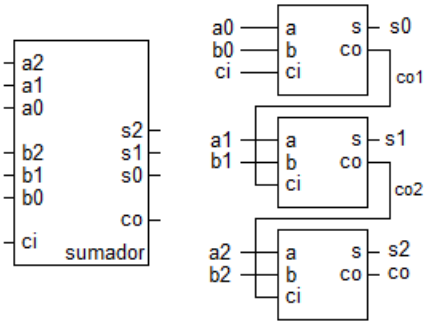
Página de Python, https://docs.python.org/3.7/index.html,
Anaconda edición individual, https://www.anaconda.com/products/individual.
Tutorial de Python en español, https://docs.python.org/es/3.7/tutorial/index.html.
matplotlib.pyplot, https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html.
Simulador de circuitos digitales, https://github.com/ArturoMigueldePriego/GRISEL.

Diagramas de circuitos lógicos

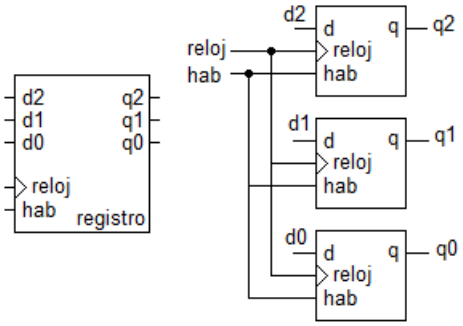
Sumador completo



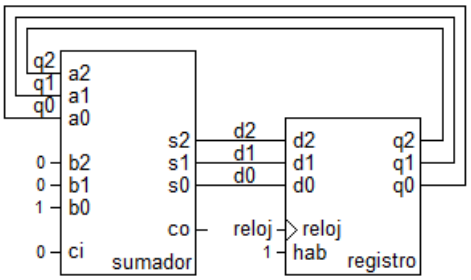
Sumador de tres bits



Registro de tres bits



Contador de tres bits



Semáforo