

# Capítulo 2: todo se vuelve programable

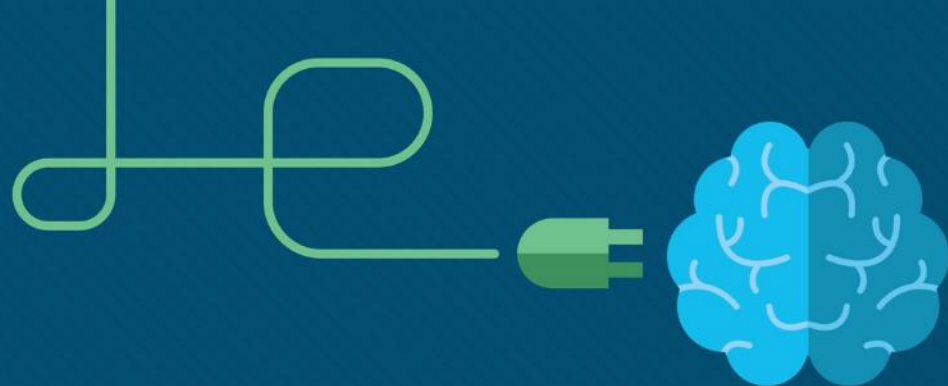
Materiales del Instructor

Introducción a Internet de las cosas v. 2.0



# Capítulo 2: todo se vuelve programable

**Introducción a Internet de las cosas  
v. 2.0: guía de planificación**



# Capítulo 2: todo se vuelve programable

Introducción a Internet de las cosas v. 2.0



# Capítulo 2: Secciones y objetivos

- 2.1 Aplicación de la programación básica para el soporte de dispositivos de IoT
  - Use Python para crear programas que acepten las entradas del usuario, y la lectura y escritura en archivos externos.
    - Describa las variables de programación básicas y aspectos esenciales.
    - Aplique las variables de programación básicas y los aspectos esenciales en Blockly.
    - Aplique las variables de programación básicas y los aspectos esenciales con Python
- 2.2 Creación de un prototipo de su idea
  - Explique la creación de prototipos y su propósito
    - Describa la creación de prototipos.
    - Describa las diversas herramientas y materiales que se usan para crear un prototipo.

## 2.1 Aplicación de la programación básica para el soporte de dispositivos de IoT

# Conceptos básicos de programación

## Siga el diagrama de flujo

Answer the following questions based on the supplied flowchart.

```
graph TD; Start([Start]) --> Read[Read door sensor]; Read --> IsOpen{Is door open?}; IsOpen -- No --> Counter0[Counter=0 Delay 2 sec]; IsOpen -- Yes --> Counter0Zero{counter=0?}; Counter0Zero -- Yes --> CounterInc[Counter=Counter+2 Delay 2 sec]; Counter0Zero -- No --> IsCounter10{Is counter >= 10?}; IsCounter10 -- Yes --> SoundAlarm[Sound Alarm]; IsCounter10 -- No --> CounterInc; CounterInc --> Read; Counter0 --> Read; SoundAlarm --> Read;
```

1. Is the sensor checking for an open or a closed door?  

Select an answer ▼

2. How frequently is the sensor checked?  

Select an answer ▼

3. Will the alarm sound if the door is open for 5 seconds?  

Select an answer ▼

4. Will the alarm sound if the door is open for 10 seconds?  

Select an answer ▼

5. Will the alarm sound if the door is open for 5 seconds, shut for 5 seconds, then reopened for 5 seconds?  

Select an answer ▼

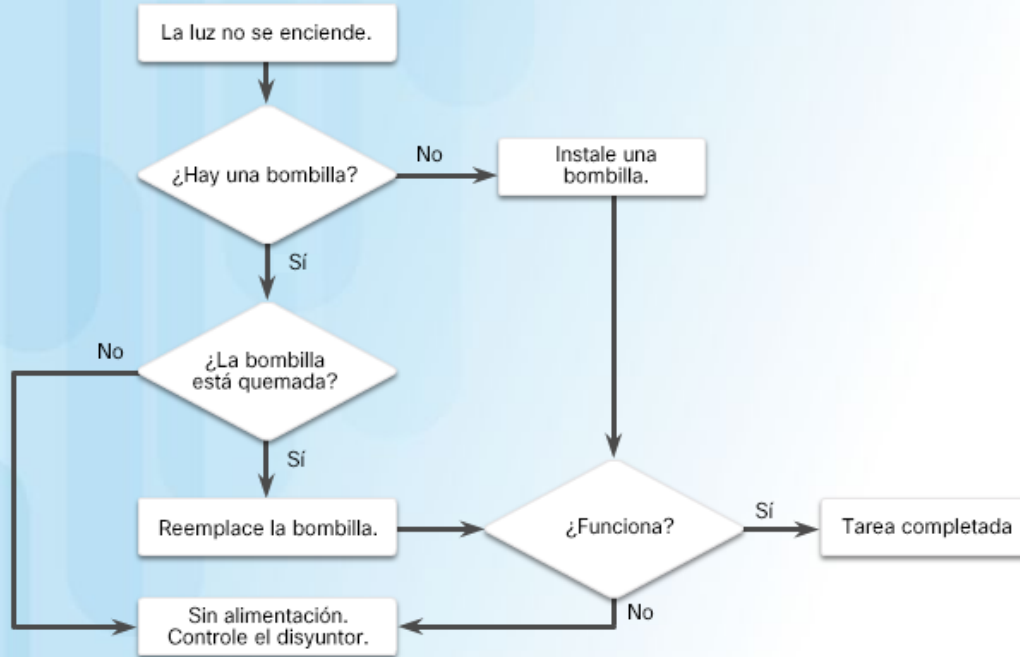
Check

Reset

# Conceptos básicos de programación

## Diagramas de flujo

Cuadro de flujo de reemplazo de bombillas



Ejemplo de un diagrama de flujo. Los rectángulos representan acciones. Los diamantes representan decisiones.

## Diagramas de flujo:

- Son diagramas que se utilizan para representar estos procesos o flujos de trabajo.
- Ilustran cómo debe funcionar un proceso.
- Muestran los estados de entrada, las decisiones tomadas y los resultados de estas.

# Software del sistema, software de aplicaciones y lenguajes informáticos

- Dos tipos comunes de software de computadora: software del sistema y el software de aplicaciones.
  - Los programas para software de aplicaciones se crean con el fin de realizar una tarea determinada o un conjunto de tareas.
  - El software del sistema funciona entre el hardware de la computadora y el programa de aplicaciones.
  - El software del sistema y el software de aplicaciones se crean con un lenguaje de programación.
  - Python es un ejemplo de un lenguaje de programación interpretado o interpretativo.

### Programa para verificar los años bisiestos en Python

```
year = int(input("Enter a year to check if it is a leap year\n"))
if (year % 4) == 0:
    if (year % 100) == 0:
        if (year % 400) == 0:
            print("{0} is a leap year".format(year))
        else:
            print("{0} is not a leap year".format(year))
    else:
        print("{0} is a leap year".format(year))
else:
    print("{0} is not a leap year".format(year))
```



# Conceptos de programación básica

## Variables de programación

- Los lenguajes de programación utilizan variables para alojar frases, números u otra información importante que pueda utilizarse en la codificación.
  - Las variables pueden contener el resultado de un cálculo, el resultado de una consulta en una base de datos o algún otro valor.
  - $x + y = z$ 
    - Aquí, "x", "y" y "z" son las variables que pueden representar caracteres, cadenas de caracteres, valores numéricos o direcciones de memorias
  - $a = 10$ 
    - asocia el valor de 10 a la variable "a"
- Las variables permiten que los programadores creen rápidamente una amplia variedad de programas simples o complejos que le indiquen a la computadora que se comporte de manera predefinida.



# Conceptos de programación básica

## Estructuras básicas de programas

```
IF (value1 > value2) THEN print_on_the_screen "Value1 is greater than Value2"
```

El código anterior emite "Value1 is greater than Value2" (El valor 1 es mayor que el valor 2) en la pantalla si la expresión Valor 1 > Valor 2 es verdadera.

```
FOR (i=0; i < 100; i++) {  
    print_on_the_screen "counter =" + i  
}
```


El código anterior emite "Counter = N" (Contador = N), donde N es el valor del contador variable "i". El mensaje se publica 100 veces en la pantalla.

```
WHILE (value < 10) {  
    print_on_the_screen "Value is still less than 10"  
    value = value + 1  
}
```

El código anterior muestra "Value is still less than 10" (El valor sigue siendo inferior a 10) en la pantalla, si Valor < 10. Tenga en cuenta que el programa también incrementa el valor cada vez que el bucle WHILE se ejecuta.

- Las estructuras lógicas más comunes son las siguientes:
  - IF – THEN:** permite que la computadora tome una decisión según el resultado de una expresión.
    - myVar > 0
    - Es verdadera si el valor almacenado en la variable myVar es mayor que cero.
    - Si es falsa, la computadora continúa con la siguiente estructura.
    - Si es verdadera, la computadora ejecuta la acción asociada antes de pasar a la siguiente instrucción del programa.
  - Los **bucles FOR** ejecutan un conjunto específico de instrucciones una cantidad de veces específica según una expresión.
    - Una variable actúa como un contador dentro de un rango de valores que se identifica con un valor mínimo y un valor máximo. Cada vez que se ejecuta el bucle, aumenta la variable del contador. Cuando el contador es igual al valor máximo definido, se abandona el bucle y la ejecución avanza a la siguiente instrucción.
  - Los **bucles WHILE** ejecutan un conjunto específico de instrucciones mientras que una expresión sea verdadera.

# Práctica de laboratorio: cree un diagrama de flujo de un proceso

 Cisco Networking Academy®Mind Wide Open™

---

### Lab – Create a Process Flowchart (Instructor Version)

**Instructor Note:** Red font color or gray highlights indicate text that appears in the instructor copy only.

#### Objectives

**Part 1: Recognize Symbols Used in a Flowchart and List Logical Process to Solve a Problem**  
**Part 2: Draw the Flowchart to Illustrate the Problem Solving Process**

#### Background

Flowcharts are diagrams used to represent processes or workflows. Using different shapes, boxes, and connecting arrows, a flowchart represents the solution flow to a given problem. Flowcharts are commonly used to represent programs, algorithms, or any ordered process in various disciplines. Flowcharts are typically created prior to starting a process or writing an application in order to verify and catch potential logic flows toward the solution before it is developed and implemented.

Flowcharts can be hand drawn or created using a number of packages including Microsoft Office products, LibreOffice, GoogleDocs, and various web applications such as <https://www.draw.io/>.

Some of the most common flowchart symbols that are used for programming are shown in the diagram along with their intended purpose for the symbol. Lines with arrows indicate the flow of the problem solving process.

Process

Data

Predefined Process

Termination

Decision

Preparation

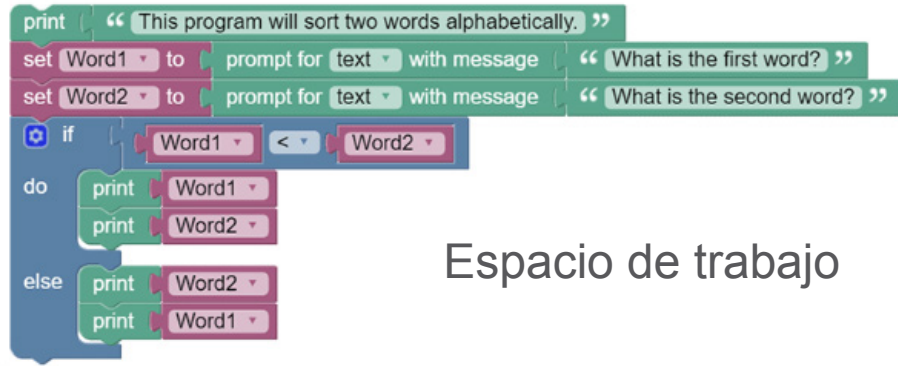
Display

Off Page Connector

Connector

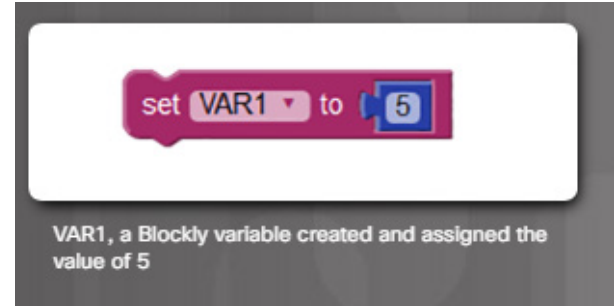
# Programación básica con Blockly

## Qué es el Blockly



Espacio de trabajo

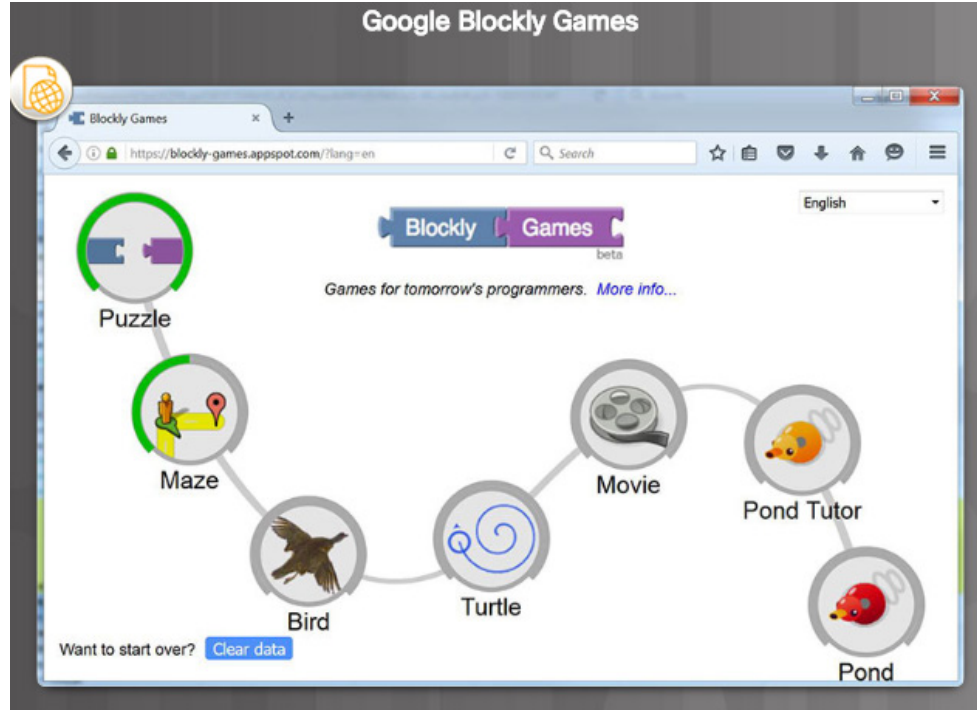
## Configuración de una variable



- Herramienta de programación visual creada para ayudar a los principiantes a comprender los conceptos de programación. Permite que un usuario cree un programa sin introducir ninguna línea de código.
- Asigna distintas estructuras de programación a los bloques de color que contienen casillas y espacios para permitir que los programadores ingresen valores. Los programadores pueden unir las estructuras arrastrando y asociando los bloques adecuados.
- Los bloques específicos representan funciones. Seleccione y arrastre los bloques de funciones hasta el espacio de trabajo y complete las casillas requeridas.

# Programación básica con Blockly

## Juegos de Blockly



<https://blockly-games.appspot.com/>

# Actividad de laboratorio – Activación de un LED con Blockly



Cisco Networking Academy®

Mind Wide Open™

## Lab – Blinking an LED Using Blockly (Instructor Version)

**Instructor Note:** Red font color or gray highlights indicate text that appears in the instructor copy only.

### Objectives

**Part 1: Open Packet Tracer and Examine Blockly Program for LED Blinking**

**Part 2: Control a RGB LED using Blockly**

### Background

Blockly is a visual programming language that lets users create programs by connecting blocks, that represent different logic language structures, rather than by writing the actual code. Blockly runs within a web browser and can translate the visually created program as JavaScript, PHP, or Python. In this lab, you will use Blockly to examine Blockly programming and to control an LED.

### Scenario

Using Blockly programming to control an IoT object LED. In this lab, Cisco Packet Tracer is used as it provides Blockly support with IoT objects.

### Required Resources

- Cisco Packet Tracer 7.1.1 and above is installed and available.

### Part 1: Launch Cisco Packet Tracer (PT) and Use Blockly

In Part 1, you will access the Cisco Packet Tracer program and examine LED control using Blockly programming.

#### Step 1: Launch Packet Tracer.

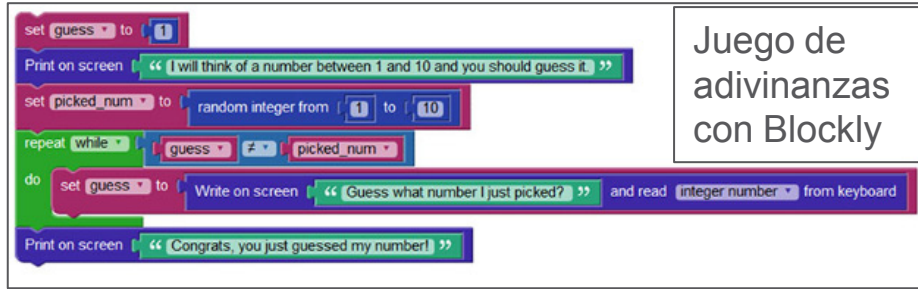
- a. Double click the Cisco Packet Tracer icon to open the PT program.



- b. The user interface is shown.

# Programación con Python

## ¿Qué es Python?



Juego de adivinanzas con Blockly

```
import random

guess = None
picked_num = None

guess = 1
print('I will think of a number between 1 and 10 and you should guess it.')
picked_num = random.randint(1, 10)
while guess != picked_num:
    guess = int(input('Guess what number I just picked? '))
print('Congrats, you just guessed my number!')
```

Juego de adivinanzas con Python

- Python es un lenguaje muy común diseñado para ser fácil de leer y escribir.
- Filosofía del lenguaje:
  - Hermoso es mejor que feo.
  - Explícito es mejor que implícito.
  - Simple es mejor que complejo.
  - Complejo es mejor que complicado.
  - La legibilidad es importante

# Programación con Python

## El intérprete de Python

```
Python 2.7 (#1, Feb 19 2010, 12:06:02)
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Mensaje de bienvenida del  
intérprete de

- El intérprete de Python comprende y ejecuta el código de Python. El código de Python pueden crearse en cualquier editor de texto y los intérpretes de Python están disponibles para muchos sistemas operativos.
- En las máquinas Linux, el intérprete de Python usualmente se instala en **/usr/bin/python** o **/usr/bin/python3**.
- Con el nuevo instalador Windows de Python, Python se instala de manera predeterminada en el directorio de inicio del usuario. Una vez instalado el intérprete de Python, funciona de manera similar al shell de Linux. Esto significa que, cuando se invoca sin argumentos, lee y ejecuta comandos interactivamente. Cuando se invoca con un argumento de nombre de archivo o con un archivo como entrada estándar, lee y ejecuta un script de ese archivo.



## El Interpretador de Python (continuación)

```
>>> the_world_is_flat = True >>> if the_world_is_flat: ... print "Be careful not to fall  
off!" ... Be careful not to fall off!
```

Bloque IF-THEN

- Para iniciar el intérprete, simplemente escriba **python** o **python3** en el indicador del shell.
- En el modo interactivo, el intérprete espera los comandos. El indicador principal está representado por tres signos mayor que (>>>). Las líneas de continuación están representadas por tres puntos (...).
- El indicador >>> indica que el intérprete está listo y espera los comandos.

# Variables y declaraciones básicas en Python

- El intérprete recibe y ejecuta las declaraciones interactivamente.

```
>>>
>>> 25+ 25
50
>>> 70 + 7*6
112
>>> (50 - 5.0*6) / 4
5.0
```

- Actúa como una calculadora simple.

```
>>>
>>> my_new_variable
Traceback (most recent call last):
  File "<stdin>", line 1, in<module>
NameError: name 'my_new_variable' is not defined
>>>
```

- Si intenta utilizar una variable no definida obtendrá como resultado un error.

```
>>>
>>> tax = 12.5 / 100
>>> price = 100.50
>>> price * tax
12.5625
>>> price + _
113.0625
>>> round(_, 2)
113.06
```

- La variable especial "\_" contiene el resultado de la última expresión publicada.

```
>>>
>>> birth_year = 1941
>>> curr_year = 2016
>>> curr_year - birth_year
75
```

- Para asignar valores a las variables, utilice el signo =.

# Variables y declaraciones básicas en Python (continuación)

- El intérprete recibe y ejecuta las declaraciones interactivamente.

```
>>>
>>> i = 256*256
>>> print ('The value of i is', i)
The value of i is 65536
```

- La declaración de publicación imprime el resultado de la expresión dada.

```
>>>
>>> 'spam eggs' # single quotes
'spam eggs'
>>> 'doesn\'t' # use \' to escape the single quote...
"doesn't"
>>> "doesn't" # ...or use double quotes instead
"doesn't"
>>> '"Yes," he said.
'"Yes," he said.
>>> "\"Yes,\" he said."
'"Yes," he said.
```

- Utilice el carácter de barra invertida (\) para sustraerse de los caracteres. Por ejemplo, una cadena que utiliza comillas dobles, pero que también necesita utilizar una comilla doble dentro de la cadena.
- Las comillas simples o comillas dobles pueden utilizarse para envolver las cadenas.

```
# Function to add two numbers:
def add_nums():
    a = 5
    b = 11
    return a+b
>>> print (add_nums())
16
>>>
```

- Las funciones permiten que un bloque de códigos reciba un nombre y se vuelva a utilizar según sea necesario.

# Funciones útiles y tipos de datos en Python

- Python admite muchas funciones y tipos de datos útiles. Algunos de los más importantes son los siguientes:

```
# One parameter
for i in range(3):
    print (i)
0
1
2
# Two parameters
for i in range(3, 6):
    print (i)
3
4
5
# Three parameters
for i in range(4, 10, 2):
    print (i)
4
6
8
```

- **Range()**: genera una lista de números utilizados generalmente para iterar con bucles FOR.
  - **range(stop)**: cantidad de números enteros a generar desde cero
  - **range ([start], stop [, paso])**: el número de Inicio de la secuencia, el número de finalización en la secuencia y la diferencia entre cada número en la secuencia.

# Funciones útiles y tipos de datos en Python (cont.)

```
tup1 = ('dancing', 'singing', 400, 1842);  
tup2 = (1, 2, 3, 4, 5, 6, 7 );  
print ('tup1[0]: ', tup1[0])  
print ('tup2[1:5]: ', tup2[1:5])
```

When the above code is executed, it produces the following result -  
tup1[0]: dancing  
tup2[1:5]: (2, 3, 4, 5)



Tuplas: secuencias separadas por paréntesis.

Listas: secuencias de objetos de Python que pueden cambiarse, y se crean configurando distintos valores separados por comas entre corchetes.



```
list1 = ['car', 'train', 47, 2016];  
list2 = [1, 2, 3, 4, 5, 6, 7 ];  
print ('list1[0]: ', list1[0])  
print ('list2[1:5]: ', list2[1:5])
```

When the above code is executed, it produces the following result -  
list1[0]: car  
list2[1:5]: [2, 3, 4, 5]

## Actualización de listas

```
list = ['car', 'train', 47, 2016];  
print ('available at index 2 : ')  
print (list[2])  
list[2] = 2017;  
print ('New value available at index 2 : ')  
print (list[2])
```

When the above code is executed, it produces the following result -  
Value available at index 2 :  
47  
New value available at index 2 :  
2017

# Funciones útiles y tipos de datos en Python (cont.)

```
x = [1,2,3,1,2,3,1,2,3]
set(x)
{1, 2, 3}
y = [1, 1, 6, 6, 6, 6, 6, 8, 8]
set(y)
{1, 6, 8}
z = [("Bird", "Cat", "Dog", "Dog", "Bird", "Bird")]
set(z)
{('Bird', 'Cat', 'Dog', 'Dog', 'Bird', 'Bird')}
```

```
animals = set(["Cow", "Fish", "Pig", "Horse"])
animals.add ("Cat")
print (animals)
set(['Fish', 'Cat', 'Horse', 'Cow', 'Pig'])

for group in [animals]:
    group.discard ("Fish")
    print (group)
set(['Cat', 'Horse', 'Cow', 'Pig'])
```

- Los conjuntos son colecciones no ordenadas de elementos exclusivos. Las aplicaciones comunes incluyen verificación de pertenencia, la eliminación de duplicados de una secuencia y cálculos de operaciones matemáticas estándar en conjuntos, como la intersección, unión, diferencia y diferencia simétrica.

# Funciones útiles y tipos de datos en Python (cont.)

## Diccionario con 4 elementos:

```
dict = {'Age' : 34, 'City' : 'Rome', 'Year' : 2016, 'Month' : 'March' }  
print ("dict['City']: ", dict['City'])  
print ("dict['Year']: ", dict['Year'])  
  
dict['City']: Rome  
dict['Year']: 2016
```

## Actualización de un valor.

```
dict['Year'] = 2015  
  
print ("dict['Year']: ", dict['Year'])  
  
dict['Year']: 2015
```

## Adición de un nuevo elemento y determinación de la cantidad de elementos en el diccionario.

```
dict['Sport'] = "Swimming"  
  
len(dict)  
  
5
```

- Un diccionario es una lista de elementos separados por comas.
- Cada elemento es una combinación de un valor y una clave única.
- Cada clave se separa de su valor por dos puntos.
- Se puede acceder a, actualizar o eliminar los elementos del diccionario.

# Estructuras de programación en Python

```
>>>
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print ('Negative changed to zero')
... elif x == 0:
...     print ('Zero')
... elif x == 1:
...     print ('Single')
... else:
...     print ('More')
...
More
```

## ▪ IF-THEN, ELSE, ELIF

- Toma decisiones según el resultado de una expresión
- ELSE especifica instrucciones para ejecutar si la expresión es falsa.
- ELIF se utiliza para realizar una segunda prueba.

```
>>>
>>> # Measure some strings:
... words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print (w, len(w))
...
cat 3
window 6
defenestrate 12
```

## ▪ Bucle FOR

- Itera los elementos de cualquier secuencia (una lista o una cadena), en el orden en que aparecen en la secuencia.


```
>>>
>>> # Fibonacci series:
... # the sum of two elements defines the next
... a, b = 0, 1
>>> while b < 10:
...     print (b)
...     a, b = b, a+b
...
1
1
2
3
5
8
```

## ▪ Bucle WHILE

- Ejecuta un bloque de código si la expresión es verdadera.



## Práctica de laboratorio: configuración de un entorno de servidores virtualizados

 Cisco Networking Academy®Mind Wide Open®

**Lab – Setting Up a Virtualized Server Environment (Instructor Version)**

**Instructor Note:** Red font color or gray highlights indicate text that appears in the instructor copy only.

**Objectives**

- Part 1: Prepare a Computer for Virtualization Environment by Installing VirtualBox
- Part 2: Install I2IoT Server Virtual Machine (VM) in VirtualBox
- Part 3: Access I2IoT Server VM
- Part 4: Basic Navigation in CentOS 7

**Background**

Computing power and resources have increased tremendously over the last 5 to 10 years. One of the benefits of having access to multicore processors and large amounts of RAM is the ability to use virtualization software to share resources. With virtualization, a user can run multiple virtual computers on one physical computer thus optimizing resource utilization.

Virtual computers that run within a physical computer are called virtual machines (VM). VMs run in an isolated environment and the activity in one VM is isolated from the activity in other VMs. The state of a VM can be saved and restored allowing one to return to a previous computing environment. VMs can be easily created, copied, and shared making them excellent tools for experimentation and prototyping.

Today, organizations use virtualized environments to host large numbers of VMs to serve the computing needs of their users. On a personal computer level, anyone with a modern computer and operating system has also the ability to run a few VMs from the desktop.

**Scenario**

In future labs, we will use the I2IoT server to learn basic computer application programming with Python. The I2IoT server is a VM hosted in a virtualization environment. A few products provide virtualization on a personal computer. For the I2IoT course, Oracle VirtualBox, a free access product, is used for the virtualized environment. In this lab, the process of downloading and installing VirtualBox will be covered. The process of adding the I2IoT server VM into the VirtualBox follows.


**Required Resources**

- A modern personal computer with sufficient RAM and with internet access.
- VirtualBox and I2IoT server are provided as downloaded images.

**Part 1: Prepare a Computer for Virtualization Environment**

In Part 1, you will download and install the virtualization software VirtualBox, a free product from Oracle.

## Práctica de laboratorio: programación básica de Python

 Cisco Networking Academy

Mind Wide Open®

---

### Lab – Basic Python Programming (Instructor Version)

**Instructor Note:** Red font color or gray highlights indicate text that appears in the instructor copy only.

#### Objectives

- Part 1: Launch VirtualBox and Enter the I2IoT server VM
- Part 2: Python Basics
- Part 3: IDLE for Python

#### Background

Python, a programming language, allows for simpler statements. Python is very easy to use, powerful, and versatile. It has become the language of choice for many IoT developers. One of the main reasons for the popularity of Python is the developer community. Python developers have created and made available many specific modules that can be imported into any program to immediately lend added functionality.

#### Scenario

In this lab, you will learn and practice some basic Python programming. More specifically, we will use Python version 3 in the lab.

#### Required Resources


- A modern personal computer with internet access and sufficient RAM.
- VirtualBox with I2IoT server installed.

### Part 1: Launch VirtualBox and Enter the I2IoT server VM

In Part 1, you launch virtualization software VirtualBox and log in to the I2IoT server VM.

#### Step 1: Launch VirtualBox.

- After VirtualBox is installed (see Lab 2.1.2.a), the VirtualBox icon should appear on the Desktop. Click the icon to launch the VirtualBox.



- Click I2IoT – GUI on the left pane to launch the server VM.

## Práctica de Laboratorio: cree un juego simple con Python



Cisco Networking Academy

Mind Wide Open™

### Lab – Create a Simple Game with Python (Instructor Version)

**Instructor Note:** Red font color or gray highlights indicate text that appears in the instructor copy only.

#### Objectives

Part 1: Launch VirtualBox and Enter the I2IoT server VM

Part 2: Create a Simple Game with Python IDLE

Part 3: IDLE for Python

#### Background

Python, a programming language, allows for simpler statements. Python is very easy to use, powerful, and versatile. It has become the language of choice for many IoT developers. One of the main reasons for the popularity of Python is the developer community; Python developers have created and made available many specific modules that can be imported into any program to immediately lend added functionality.

#### Scenario

In this lab, you will create a simple game using Python IDLE.

#### Required Resources

- A modern personal computer with sufficient RAM and with internet access.
- VirtualBox with I2IoT server installed.

#### Part 1: Launch VirtualBox and Enter the I2IoT server VM

In Part 1, you launch virtualization software VirtualBox and login to the I2IoT server VM.

##### Step 1: Launch VirtualBox.

- After VirtualBox is installed (see Lab 2.1.2.a), the VirtualBox icon should appear on the Desktop. Click the icon to launch the VirtualBox.



- Click I2IoT – GUI on the left pane to launch the server VM.

## 2.2 Creación de un prototipo de su idea

¿Qué es la creación de un prototipo?

## Definición de creación de un prototipo

- La creación de prototipos es el proceso de creación de un modelo de trabajo de un producto o sistema.
- En IoT, esto ayuda a tener habilidades de diseño, habilidades eléctricas, habilidades físicas/mecánicas, habilidades de programación y a comprender cómo funciona TCP/IP.
- Debido a que IoT aún está en desarrollo, todavía existen tareas desconocidas por descubrir.
- Este es un gran momento para inventar algo que forme parte de IoT.

- Tiene plena capacidad de funcionamiento, pero no de prueba de fallas.
- Es una versión real, con capacidad de funcionamiento del producto.
- Se utiliza para evaluar el rendimiento y mejorar el producto.
- La parte interna y la parte externa están completas.
- Su producción es relativamente costosa.
- En IoT se suele utilizar para demostrar la tecnología.

¿Qué es la creación de un prototipo?

## Cómo crear un prototipo



- ¿Cómo se crea un prototipo? Un equipo de Google utilizó el “método rápido de creación de un prototipo” para crear Google Glass.
- Kickstarter, Indiegogo y Crowdfunder son solo tres de los numerosos programas de financiación colectiva en línea.
- ¿Qué invención de IoT creará?

# Materiales físicos



- Un buen lugar para comenzar es, por supuesto, Internet. Personas que nunca se encontraron físicamente ahora puedan colaborar y trabajar juntas.
- Maker Media es una plataforma global para conectar a los creadores entre sí a fin de que intercambien proyectos e ideas.
- Making Society tiene una buena sección sobre la creación de modelos de plástico y arcilla.
- LEGO Mindstorms tiene una gran comunidad de colaboradores y fanáticos.
- Meccano, o Erector Set, es un sistema de construcción de modelos que consiste en bandas de metal, placas, vigas angulares, ruedas, ejes y engranajes, todos reutilizables, con tuercas y pernos para conectar las piezas. Le permite armar prototipos funcionales y dispositivos mecánicos.
- La impresión en 3D es el proceso por el cual se crea un objeto sólido basado en un archivo informático de un modelo en 3D.

# Kits de herramientas electrónicas

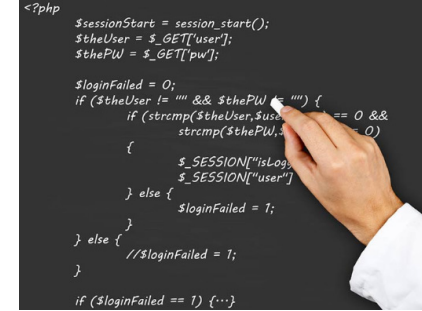


- Aunque es posible crear programas para casi cualquier computadora, algunas plataformas están diseñadas para principiantes. A continuación, se indican algunas de las plataformas más populares:
  - Arduino es una plataforma informática física de código abierto basada en una simple placa de microcontroladores y un entorno de desarrollo para escribir el software de la placa. Se pueden desarrollar objetos interactivos que recogen información de diversos switches o sensores para controlar luces, motores y otros objetos físicos.
  - Raspberry Pi es una computadora de bajo costo del tamaño de una tarjeta de crédito que se conecta a un monitor de computadora o a un televisor. Se opera mediante un teclado y un mouse estándar. Es capaz de funcionar como una computadora, desde navegar en Internet y reproducir video de alta definición, hasta crear hojas de cálculo, procesar texto y usar juegos.
  - Beaglebone es muy similar a Raspberry Pi en tamaño, requisitos de energía y aplicación. Beaglebone tiene más capacidad de procesamiento que Raspberry Pi, por lo tanto es una mejor opción para aplicaciones con mayores requisitos de procesamiento.



# Recursos para la creación de prototipos

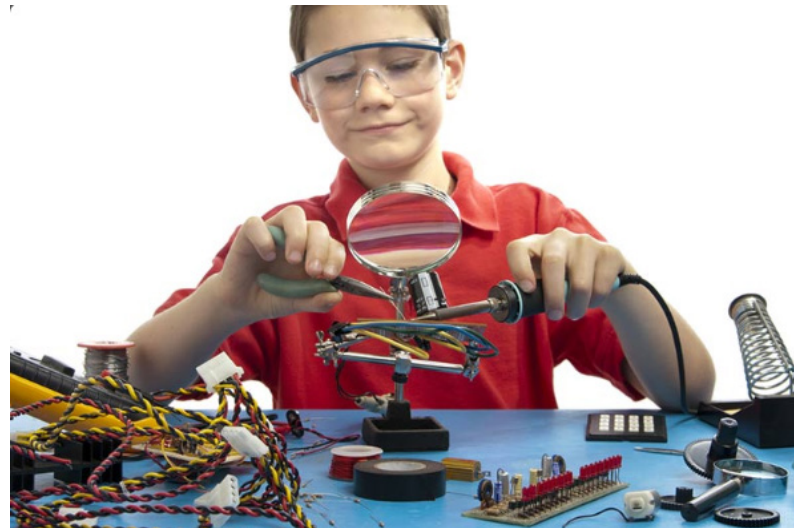
## Recursos de programación




- La programación es fundamental para IoT. La creación de un código personalizado es muy útil cuando se desarrolla una solución de IoT. Existen muchos otros recursos gratuitos que pueden ayudarlo a aprender sobre programación:
- El MIT OpenCourseWare (OCW) es una publicación basada en la Web de casi todo el contenido de los cursos de MIT. Abierto y disponible para todo el mundo, OCW es un excelente lugar para familiarizarse con la programación informática de manera gratuita. Se pueden buscar cursos de OCW relacionados con programación en <http://ocw.mit.edu/courses/intro-programming>.
- Khan Academy es un sitio web educativo sin fines de lucro creado en 2006 para proporcionar “educación libre, de primera clase, a cualquier persona y en cualquier lugar”. Las lecciones relacionadas con la programación informática se pueden encontrar en <https://www.khanacademy.org/computing/cs>
- Code Academy es otro excelente recurso. Se basa en la interactividad para enseñar a las personas a escribir programas informáticos. Los puede encontrar en <http://www.codecademy.com>.

# Talleres de invención y de emprendimiento en la comunidad

- Entonces, es probable que haya creado algo realmente genial. ¿Qué hacer ahora? Existen varios lugares en los que puede recibir ayuda para exponer su idea o su prototipo a otros.
- Investigue cuáles son las opciones disponibles en su comunidad.
- Internet tiene muchos recursos para ayudar a que su idea tenga exposición. Quirky es un buen ejemplo. Quirky permite que los usuarios compartan sus ideas. Cuando se envía una idea, otros usuarios de Quirky pueden votar y elegir si desean apoyarla o no. Si una idea es buena, se puede convertir en un producto real. Puede conocer más acerca de Quirky en <https://www.quirky.com/how-it-works>.




# Práctica de laboratorio: configuración de PL-App con Raspberry Pi

**CISCO** Cisco Networking Academy<sup>®</sup>Mind Wide Open<sup>™</sup>

### Lab: Setting up PL-App with a Raspberry Pi

#### Lab Topology



#### Objectives

- Set up a Raspberry Pi board as a PL-App device
- Use PL-App Launcher to provision and discover PL-App devices

#### Background

Cisco Prototyping Lab is a set of hardware and software components that enable students and instructors to learn about, to prototype, and to model various IoT, digitization and data analytics solutions.


The hardware components are part of the Prototyping Lab Kit (PL-Kit). The PL-Kit is based on Open HW prototyping boards such as Raspberry Pi and Arduino and includes additional sensors, actuators, and electronic components. The PL-Kit can be used to build sophisticated prototypes of end to end IoT systems that can sense and actuate the real physical world, analyze and process the data at the fog layer, and connect to network and cloud systems.

The primary software component of the Prototyping Lab is the Prototyping Lab App (PL-App). The PL-App is a software platform running on a Raspberry Pi that exposes a web interface based on a concept of notebooks. A notebook is an interactive web page where content is distributed in what are called cells. The first cell type is called Markdown and is a cell that contains standard objects such as text, images, videos, etc. The second cell type is called Code cell and is a cell with executable code of different programming languages (the default is Python).

A notebook can be used as a lab where the explanatory text is placed with executable code and together create a scaffolded learning experience. The explanatory text guides the student through the learning experience, while hands on skills are acquired by modifying, examining and executing executable code.

A notebook is also a great tool that can be used to prototype IoT systems, interconnect with existing cloud services using APIs, etc. In a notebook, application code can be split between multiple code cells, executing only the part of the code that is just being developed or troubleshot. Moreover, using markdown cells, documentation and explanatory text can be added between code cells to provide a clean, easy to understand Rapid Prototyping Interface.

# Práctica de laboratorio opcional: uso de una notebook de PL-App

**cisco**


Cisco Networking Academy®

Mind Wide Open®

---

### Lab: Using a PL-App Notebook

#### Lab Topology



The diagram illustrates the lab topology. A central cloud labeled 'LAN' connects several components. On the left, a 'PL-App' is shown with a user icon. A red arrow labeled 'Board Provisioning', 'List of Devices', and 'Login to Devices' points from the PL-App to a 'PL-App Launcher' window. The launcher window displays a list of devices. An orange arrow labeled 'Discovery of devices' points from the launcher to a 'PL-App Image' and a 'PL-Kit' (represented by a green circuit board). A green arrow labeled 'Jupyter Notebooks: Labs, Code examples, Dev environment' points from the PL-App to the PL-Kit. The PL-App Image is also connected to the LAN cloud.

#### Objectives

- Access existing PL-App notebooks
- Learn how to interact with PL-App notebooks using keyboard shortcuts
- Create your own PL-App notebook

#### Background

The PL-App notebook concept is based on the Jupyter open source project that provides an interactive web page, where the content is divided into multiple smaller sections called cells. Each cell can contain either Markdown type of code or executable code.

The Markdown cell type is used to write the standard text of the page. Besides formatted text, it can hold multimedia objects like pictures, videos, animations, etc., and it is used to introduce and explain learning objectives, or is used as part of the documentation.

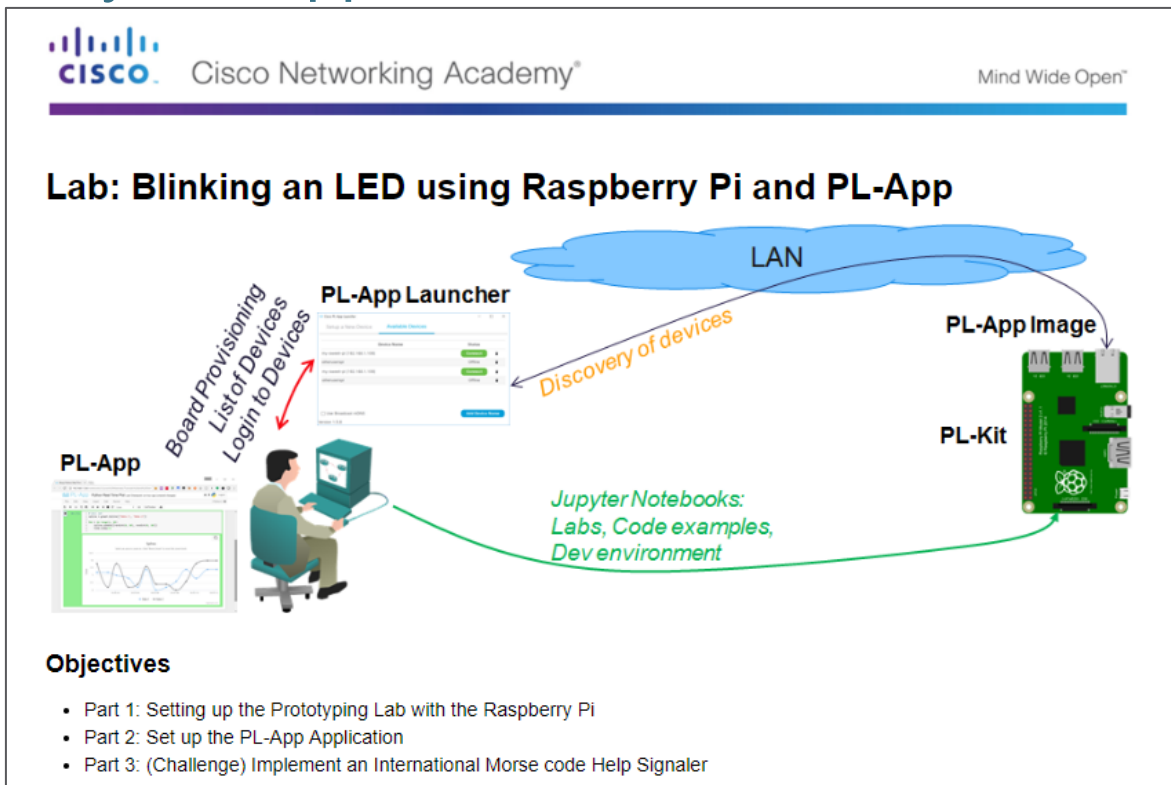
The Code cells are used to write directly executable code in one of the supported programming languages (Python, Bash). The application code can be split into multiple cells, where each cell can be executed one by one, multiple times. The code in these cells can be directly edited and modified, re-executed multiple times, and adapted to specific needs. This enables rapid prototyping concepts, where the development of the final application can be divided into smaller sections, each dealing only with the specific problem, while all the cells are solving the overall problem.

**cisco**


es. Todos los derechos reservados. Información confidencial de Cisco.

43

# Práctica de laboratorio opcional: activación de un LED con Raspberry Pi y PL-App



# Práctica de laboratorio opcional: introducción a Arduino

 Cisco Networking Academy<sup>®</sup>Mind Wide Open<sup>™</sup>

---

## Challenge Lab – Introduction to Arduino (Instructor Version)

**Instructor Note:** Red font color or gray highlights indicate text that appears in the instructor copy only.

### Objectives

**Part 1: Installing the Arduino IDE Software**

**Part 2: Using the Arduino IDE Software**

### Background / Scenario

Arduino is a prototyping platform that allows users to create programs to control hardware. In this lab, you will learn to use the Arduino and Arduino IDE to control the blinking rate of an LED.

### Required Resources

- Arduino Redboard or Uno
- A USB cable for connection to the PC
- 1 LED

**Note:** The challenge labs in this course assume that the student has all of the necessary hardware to perform them. If you do not have the necessary hardware and wish to complete these labs, you may wish to purchase kits which contain all of the hardware for the challenge labs and additional hardware which can be used to complete additional experiments beyond this course. Make sure to read the required resources for each challenge lab to understand what hardware is required.

### Part 1: Installing the Arduino IDE Software

#### Step 1: Download the software.

- Navigate to <https://www.arduino.cc/en/Main/Software>. Select **Windows Installer** on the right panel to download the software if you have administrative rights to your computer. Click **JUST DOWNLOAD** and click **Save File** and save it to the Downloads folder.
- When the download is finished, navigate to the location where the file has been downloaded. Open the **Downloads** folder.

#### Step 2: Install the software.

- Install Arduino by opening the arduino-x.x.x windows.exe file, where x represents the version number. Click **Yes** in the User Account Control dialog box, if necessary. Click **I Agree** to continue the installation and follow the on-screen instructions to finish the installation.

## 2.3 Resumen del capítulo

# Resumen

- Los diagramas de flujo son diagramas que se utilizan para representar procesos.
- Existen dos tipos comunes de software informático: software del sistema y software de aplicaciones. Los programas para software de aplicaciones se crean con el fin de realizar una tarea determinada. El software del sistema funciona entre el hardware de la computadora y el programa de aplicaciones.
- Las estructuras lógicas más comunes son los bucles IF–THEN, FOR y WHILE.
- Blockly es una herramienta de programación visual creada para ayudar a los principiantes a comprender los conceptos de programación. Blockly implementa la programación visual mediante la asignación de diferentes estructuras de programas a bloques de color.
- Python es un lenguaje muy común diseñado para ser fácil de leer y escribir. Python es un lenguaje interpretado; por lo tanto, requiere un intérprete para analizar y ejecutar el código de Python.
- Python admite muchas funciones y tipos de datos, incluidos **Range()**, **tuplas**, **listas**, **conjuntos**, **diccionario**. Python también implementa dos subestructuras denominadas ELSE y ELIF.



