

Everything Becomes Programmable

Introduction to the Internet of Things v2.0



Sections & Objectives

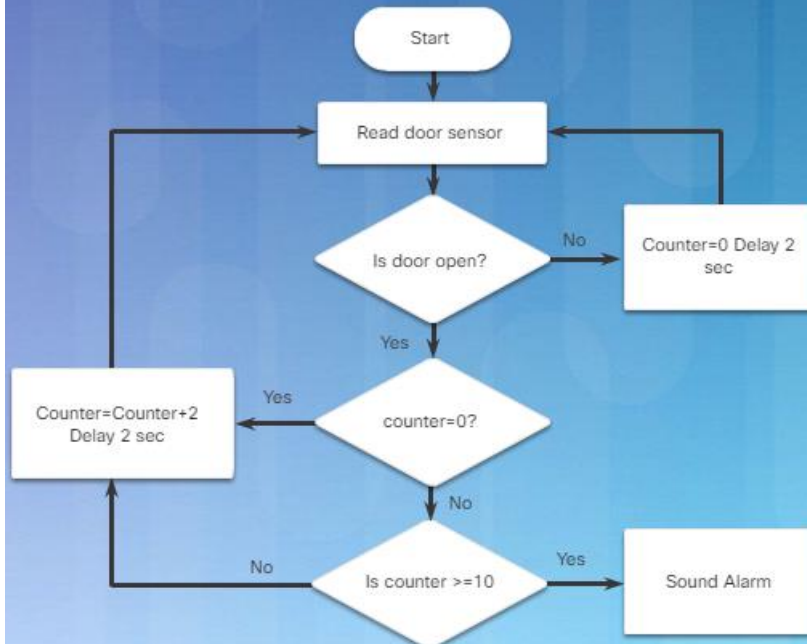
- Apply Basic Programming to Support IoT Devices
 - Use Python to create programs that accept user input and read and write to external files.
 - Describe basic programming variables and fundamentals.
 - Apply basic programming variables and fundamentals in Blockly.
 - Apply basic programming variables and fundamentals using Python
- Prototyping Your Idea
 - Explain prototyping and its purpose
 - Describe Prototyping.
 - Describe the various tools and materials to use to prototype.

Apply Basic Programming to Support IoT Devices

Basic Programming Concepts

Follow the Flowchart

Answer the following questions based on the supplied flowchart.



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 ▼

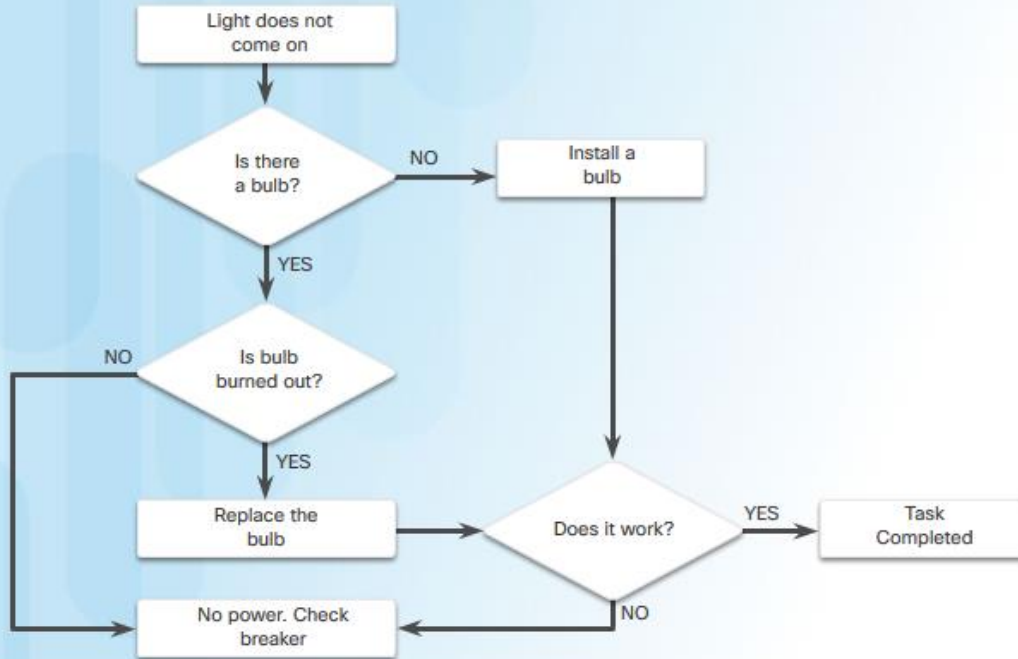
Check

Reset

Basic Programming Concepts

Flowcharts

Light Bulb Replacement Flow Chart



"Example of a flow chart. Rectangles represent actions. Diamonds represent decisions"

Flowcharts:

- Diagrams that are used to represent processes or workflows.
- Illustrate how a process should work.
- Show input states, any decisions made, and the results of those decisions.

System Software, Application Software, and Computer Languages

- Two common types of computer software: system software and application software.
 - Application software programs are created to accomplish a certain task or collection of tasks.
 - System software works between the computer hardware and the application program.
 - Both system software and application software are created using a programming language.
 - Python is an example of an interpreted programming language.

Program to Verify Leap Years in 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))
```

Basic Programming Concepts

Programming Variables

- Programming languages use variables to hold phrases, numbers, or other important information that can be used in coding.
 - Variables can hold the result of a calculation, the result of a database query, or some other value.
 - $x + y = z$
 - “x, y and z” are variables which can represent characters, character strings, numeric values or memory addresses
 - $a = 10$
 - associates the value 10 to variable “a”
- Variables allow programmers to quickly create a wide range of simple or complex programs which tell the computer to behave in a pre-defined fashion.



Basic Programming Concepts

Basic Program Structures

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

The code above prints "Value1 is greater than Value2" on the screen, if the expression `value1 > value2` is true.

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

The code above prints "counter = N" (where N is the value of the counter variable "i".)
The message is printed 100 times on the screen.

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

The code above prints "Value is still less than 10" on the screen while `value < 10`. Notice that the program also increments `value` every time the WHILE loop is executed.

- Most common logic structures are:
 - **IF – THEN** allows the computer to make a decision based on the result of an expression.
 - `myVar > 0`
 - True if the value stored in the `myVar` variable is greater than zero.
 - If false, the computer moves on to the next structure,
 - If true, the computer executes the associated action before moving on to the next instruction in the program.
 - **FOR Loops** execute a specific set of instructions a specific number of times, based on an expression.
 - A variable acts as a counter inside a range of values identified by a minimum and a maximum. Every time the loop is executed, the counter variable is incremented. When the counter is equal to the defined maximum value, the loop is abandoned and the execution moves on to the next instruction.
 - **WHILE Loops** execute a specific set of instructions while an expression is true.

Lab – Create a Process Flowchart



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

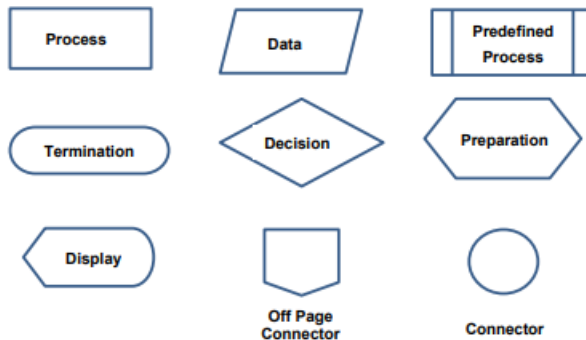
Part 2: Draw the Flowchart to Illustrate the Problem Solving Process

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.

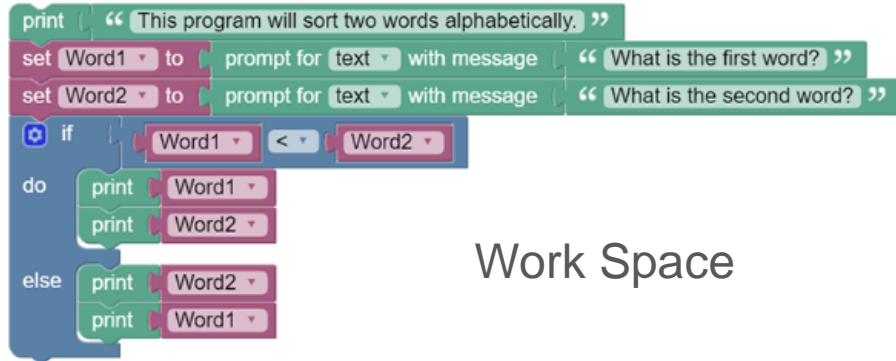
Flowcharts can be hand drawn or created using a number of packages including Microsoft Office products, LibreOffice, GoogleDocs, and various web applications such as <https://www.draw.io/>.

Some of the most common flowchart symbols that 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.



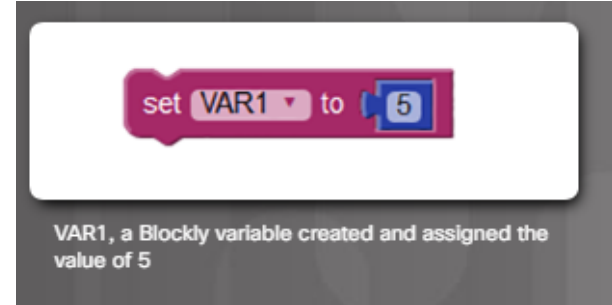
Basic Programming Using Blockly

What is Blockly



Work Space

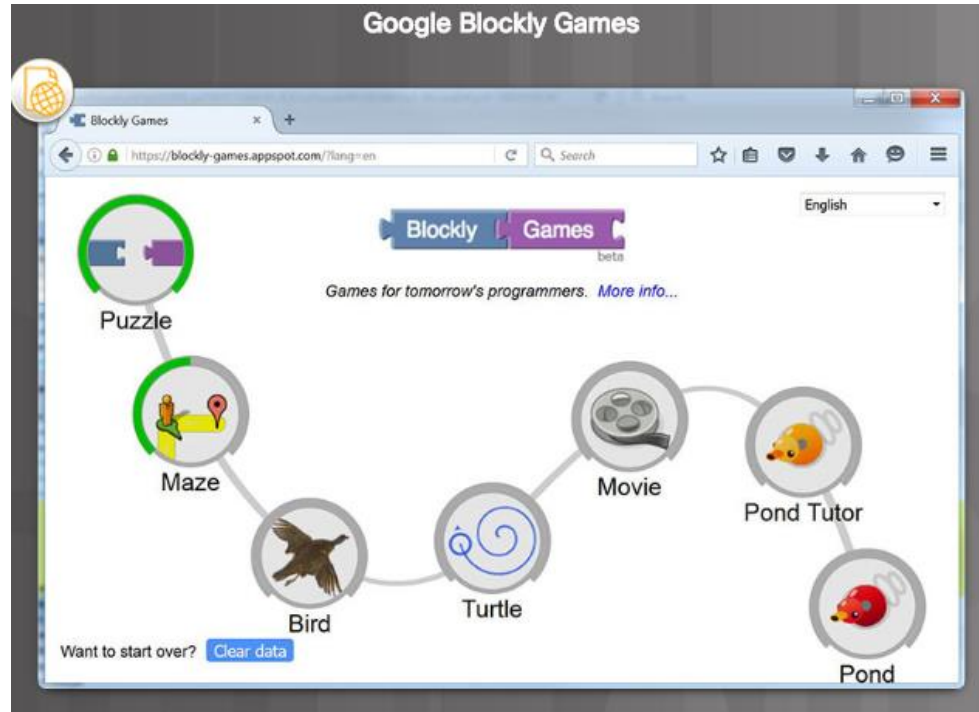
Setting Up a Variable



- Visual programming tool created to help beginners understand the concepts of programming. Allows a user to create a program without entering any lines of code.
- Assigns different programming structures to colored blocks which contain slots and spaces to allow programmers to enter values. Programmers can connect structures together by dragging and attaching the appropriate blocks.
- Specific blocks represent functions. Select and drag function blocks to the work space and fill in the required slots.

Basic Programming Using Blockly

Blockly Games



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

Basic Programming Using Blockly

Lab – Blinking an LED Using 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.

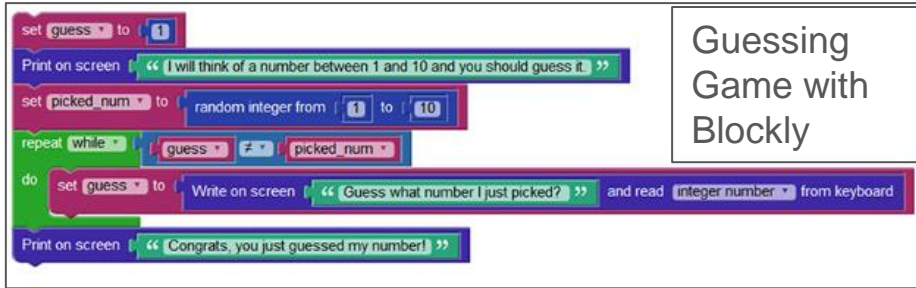
- a. Double click the Cisco Packet Tracer icon to open the PT program.



- b. The user interface is shown.

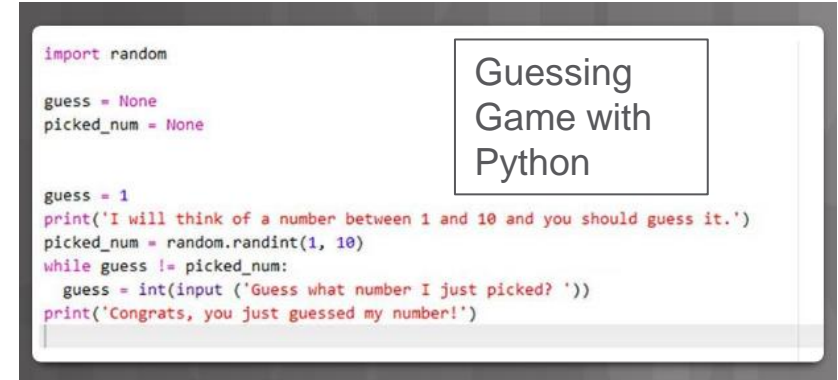
Programming with Python

What is Python?



Guessing Game with Blockly

The image shows a Blockly script for a guessing game. The code starts by setting a variable 'guess' to 1. It then prints a message: "I will think of a number between 1 and 10 and you should guess it." Next, it sets a variable 'picked_num' to a random integer between 1 and 10. A 'repeat while' loop is used to check if 'guess' is not equal to 'picked_num'. Inside the loop, it prompts the user to "Guess what number I just picked?" and reads an integer from the keyboard. After the loop, it prints "Congrats, you just guessed my number!"



Guessing Game with Python

```
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!')
```

The image shows a Python script for a guessing game. It imports the 'random' module. It initializes 'guess' to None and 'picked_num' to None. It sets 'guess' to 1 and prints a message: "I will think of a number between 1 and 10 and you should guess it." It then uses 'random.randint(1, 10)' to pick a number. A 'while' loop checks if 'guess' is not equal to 'picked_num'. Inside the loop, it prompts the user to "Guess what number I just picked?" and reads an integer from the keyboard. After the loop, it prints "Congrats, you just guessed my number!"

- Python is a very popular language that is designed to be easy to read and write.
- Philosophy of the language:
 - Beautiful is better than ugly
 - Explicit is better than implicit
 - Simple is better than complex
 - Complex is better than complicated
 - Readability counts

Programming with Python

The Python Interpreter

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

Interpreter Welcome Message

- The Python interpreter understands and executes Python code. Python code can be created in any text editor and Python interpreters are available for many operating systems.
- In Linux machines, the Python interpreter is usually installed in **`/usr/bin/python`** or **`/usr/bin/python3`**.
- With the new Windows Python installer, Python is installed by default into the user's home directory. After the Python interpreter has been installed, it operates somewhat like the Linux shell. This means that when called with no arguments, it reads and executes commands interactively. When called with a file name argument or with a file as standard input, it reads and executes a script from that file.

The Python Interpreter (Cont.)

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

IF-THEN Block

- To start the interpreter, simply type **python** or **python3** at the shell prompt.
- In interactive mode, the interpreter waits for commands. The primary prompt is represented by three greater-than signs (>>>). Continuation lines are represented by three dots (...).
- The >>> prompt indicates the interpreter is ready and waiting commands.

Variables and Basic Statements in Python

- The interpreter receives and executes statements interactively.

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

- Acts as a simple calculator.

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

- Special variable “_” holds the result of the last expression issued.

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

- Attempts to use a not defined variable will result in an error.

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

- To assign values to variables, use the = sign.

Variables and Basic Statements in Python (Cont.)

- The interpreter receives and executes statements interactively.

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

- Print statement prints the result of the expression it was given.

```
>>>
>>> '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.
```

- Use the backslash character (\) to escape characters. As an example, a string uses double quotes but also needs to use a double quote within the string.
- Single quotes or double quotes can be used to wrap strings.

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

- Functions allow for a block of code to be given a name and re-used as needed.

Useful Functions and Data Types in Python

- Python supports many useful functions and datatypes. Some of the more important ones are as follows:

```
# 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()** - Generates a list of numbers usually used to iterate with FOR loops.
 - range(stop)** - number of integers (whole numbers) to generate, starting from zero
 - range([start], stop[, step])** – Starting number of the sequence, the ending number in the sequence, and the difference between each number in the sequence.

Useful Functions and Data Types in 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)



Tuples - sequences, separated by parentheses.

Lists - sequence of changeable Python objects, created by putting different comma-separated values between square brackets.



```
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]

Updating Lists

```
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

Useful Functions and Data Types in 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'])
```

- Sets are unordered collections of unique elements. Common uses include membership testing, removing duplicates from a sequence, and computing standard math operations on sets such as intersection, union, difference, and symmetric difference.

Useful Functions and Data Types in Python (Cont.)

Dictionary with 4 elements:

```
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
```

Update a value

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

dict['Year']: 2015
```

Add a new element and determine the number of elements in the dictionary

```
dict['Sport'] = "Swimming"

len(dict)

5
```

- A dictionary is a list of elements that are separated by commas.
- Each element is a combination of a value and a unique key.
- Each key is separated from its value by a colon.
- Dictionary elements can be accessed, updated, and deleted.

Programming Structures in 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

- Make decisions based upon the result of an expression
- ELSE specify instructions to be executed if the expression is false.
- ELIF is used to perform a second test.

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

▪ FOR Loop


- Iterates the items of any sequence (a list or a string), in the order that they appear in the sequence

```
>>>
>>> # 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
```

▪ WHILE Loop

- Executes a block of code if the expression is true

Lab – Setting Up a Virtualized Server Environment



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.

Programming with Python

Lab – Basic Python Programming

 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.

Lab – Create a Simple Game with Python



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.

Prototyping Your Idea

What is Prototyping?

Defining Prototyping

- Prototyping is the process of creating a working model of a product or system.
- In IoT, it helps to have design skills, electrical skills, physical/mechanical skills, programming skills, and to understand how TCP/IP works.
- Because the IoT is still developing, there are still unknown tasks to discover.
- This is a great time to invent something that is part of the IoT.

- Is fully functional, but not fault-proof.
- Is an actual, working version of the product.
- Is used for performance evaluation and further improvement of product.
- Has a complete interior and exterior.
- May be relatively expensive to produce.
- In the IoT, is often used as a technology demonstrator.

What is Prototyping?

How to Prototype



- How do you prototype? A team at Google used the “Rapid Prototyping Method” to create the Google Glass.
- Kickstarter, Indiegogo, and Crowdfunder are just three of the many online crowd funding programs.
- What IoT invention will you create?

Prototyping Resources

Physical Materials



- A good place to start is, of course, the Internet. People who have never physically met can now collaborate and work together.
- Maker Media is a global platform for connecting makers with each other to exchange projects and ideas.
- Making Society has a good section on modeling plastic and clay.
- LEGO Mindstorms has a large community of contributors and fans.
- Meccano, or Erector Set, is a model construction system that consists of reusable metal strips, plates, angle girders, wheels, axles, and gears, with nuts and bolts to connect the pieces. It lets you build working prototypes and mechanical devices.
- 3D printing is the process of making a solid object based on a 3D model computer file.

Prototyping Resources

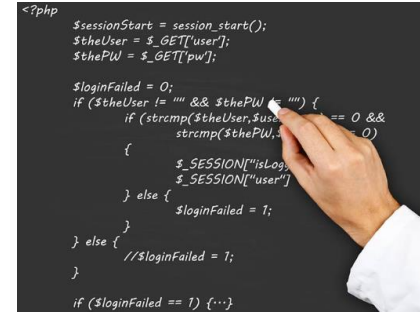
Electronic Toolkits



- While you can create programs for almost any computer, some platforms are designed for the beginner. Below you will find some of the most popular platforms:
 - Arduino is an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. You can develop interactive objects that take input from a variety of switches or sensors to control lights, motors, and other physical objects.
 - Raspberry Pi is a low cost, credit-card-sized computer that plugs into a computer monitor or TV. You operate it using a standard keyboard and mouse. It is capable of doing everything a computer can do, from browsing the Internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.
 - The Beaglebone is very similar to the Raspberry Pi in size, power requirements, and application. The Beaglebone has more processing power than the Raspberry Pi; therefore, it is a better choice for applications with higher processing requirements.

Prototyping Resources

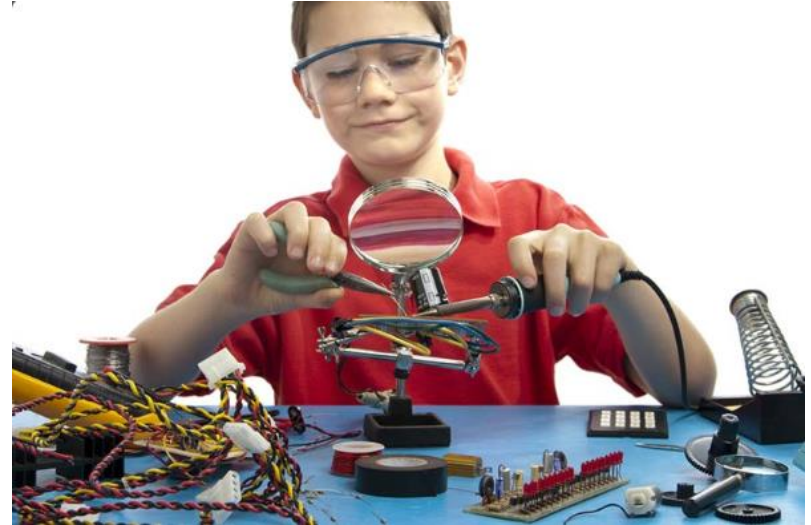
Programming Resources




- Programming is critical to the IoT. Creating custom code is very useful when developing an IoT solution. There are many other free resources that can help you get started with programming:
 - The MIT OpenCourseWare (OCW) is a web-based publication of almost all MIT course content. Open and available to the world, OCW is a great place to get familiar with computer programming for free. OCW programming related courses can be found at <http://ocw.mit.edu/courses/intro-programming>.
 - Khan Academy is a non-profit educational website created in 2006 to provide “a free, world-class education for anyone, anywhere”. The lectures related to computer programming can be found at <https://www.khanacademy.org/computing/cs>.
 - Code Academy is another excellent resource. It relies on interactivity to help people learn how to write computer programs. You can find them at <http://www.codecademy.com>.

Community Inventor and Entrepreneurship Workshops

- So, perhaps you have just created something really great. What now? There are a number of places where you can get help exposing your idea or prototype to others.
- Investigate what is available in your community.
- The Internet has many resources to help your idea get exposure. A good example is Quirky. Quirky allows users to share their ideas. When an idea is submitted, other Quirky users can vote and choose whether or not they want to support your idea. If an idea is good, it may become a real product. You can learn more about Quirky at <https://www.quirky.com/how-it-works>.




Optional Lab – Setting up PL-App with the Raspberry Pi

 Cisco Networking Academy[®] Mind Wide Open[™]

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.

Optional Lab – Using a PL-App Notebook

 Cisco Networking Academy[®] Mind Wide Open[™]

Lab: Using a PL-App Notebook

Lab Topology



The diagram illustrates the lab topology. On the left, a person is shown at a computer workstation. A red arrow labeled 'Board Provisioning List of Devices Login to Device' points from the workstation to the 'PL-App Launcher' interface. The 'PL-App Launcher' is connected to a 'LAN' cloud. An orange arrow labeled 'Discovery of devices' points from the LAN to a 'PL-App image' and a 'PL-Kit' (represented by a green circuit board). A green arrow labeled 'Jupyter Notebook: Labs, Code examples, Dev environment' points from the PL-App image back to the workstation.

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.



© 2016 Cisco and/or its affiliates. All rights reserved. Cisco Confidential 34

Optional Lab – Blinking an LED using Raspberry Pi and PL-App

Cisco Networking Academy®

Mind Wide Open™

Lab: Blinking an LED using Raspberry Pi and PL-App

The diagram illustrates the lab setup. A user is shown at a computer interacting with the **PL-App Launcher**. A red arrow labeled "Board Provisioning List of Devices Login to Devices" points from the user to the launcher. The launcher is connected to a **LAN** cloud. An orange arrow labeled "Discovery of devices" points from the LAN to the **PL-App Image**. A green arrow labeled "Jupyter Notebooks: Labs, Code examples, Dev environment" points from the **PL-App Image** to the **PL-Kit** (Raspberry Pi).

Objectives

- Part 1: Setting up the Prototyping Lab with the Raspberry Pi
- Part 2: Set up the PL-App Application
- Part 3: (Challenge) Implement an International Morse code Help Signaler

Optional Lab – Introduction to Arduino



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.

Chapter Summary

Chapter Summary

Summary

- Flowcharts are diagrams that are used to represent processes.
- There are two common types of computer software: system software and application software. Application software programs are created to accomplish a certain task. System software works between the computer hardware and the application program.
- The most common logic structures are IF – THEN, FOR Loops, and WHILE Loops.
- Blockly is a visual programming tool created to help beginners understand the concepts of programming. Blockly implements visual programming by assigning different programming structures to colored blocks.
- Python is a very popular language that is designed to be easy to read and write. Python is an interpreted language; therefore, an interpreter is required to parse and execute Python code.
- Python supports many useful functions and datatypes including **Range()**, **Tuples**, **Lists**, **Sets**, **Dictionary**. Python also implements two sub-structures named ELSE and ELIF.

