In [3]:
```
print("Hello World")
```

```
Hello World
```

In [4]:
```
#Import the base overlay, rgbleds, leds
from pynq.overlays.base import BaseOverlay
import time
base = BaseOverlay("base.bit")
```

In [5]:
```
help(base)
```

```
Help on BaseOverlay in module pynq.overlays.base.base object:

class BaseOverlay(pynq.overlay.Overlay)
 |  The base overlay for the Pynq-Z2
 |
 |  This overlay is designed to interact with all of the on board peripherals
 |  and external interfaces of the Pynq-Z2 board. It exposes the following
 |  attributes:
 |
 |  Attributes
 |  ----------
 |  iop_pmoda : IOP
 |      IO processor connected to the PMODA interface
 |  iop_pmodb : IOP
 |      IO processor connected to the PMODB interface
 |  iop_arduino : IOP
 |      IO processor connected to the Arduino interface
 |  iop_rpi : IOP
 |      IO processor connected to the RPi interface
 |  trace_rpi : pynq.logictools.TraceAnalyzer
 |      Trace analyzer block on RPi interface, controlled by PS.
 |  trace_pmoda : pynq.logictools.TraceAnalyzer
 |      Trace analyzer block on PMODA interface, controlled by PS.
 |  trace_pmodb : pynq.logictools.TraceAnalyzer
 |      Trace analyzer block on PMODB interface, controlled by PS.
 |  leds : AxiGPIO
 |      4-bit output GPIO for interacting with the green LEDs LD0-3
 |  buttons : AxiGPIO
 |      4-bit input GPIO for interacting with the buttons BTN0-3
 |  switches : AxiGPIO
 |      2-bit input GPIO for interacting with the switches SW0 and SW1
 |  rgbleds : [pynq.board.RGBLED]
 |      Wrapper for GPIO for LD4 and LD5 multicolour LEDs
 |  video : pynq.lib.video.HDMIWrapper
 |      HDMI input and output interfaces
 |  audio : pynq.lib.audio.Audio
 |      Headphone jack and on-board microphone
 |  pin_select : GPIO
 |      The pin selection between PMODA (0) and RPI header (1).
 |
 |  Method resolution order:
 |      BaseOverlay
 |      pynq.overlay.Overlay
 |      pynq.bitstream.Bitstream
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, bitfile, **kwargs)
 |      Return a new Overlay object.
 |
 |      An overlay instantiates a bitstream object as a member initially.
 |
 |      Parameters
 |      ----------
 |      bitfile_name : str
 |          The bitstream name or absolute path as a string.
 |      dtbo : str
 |          The dtbo file name or absolute path as a string.
 |      download : bool
 |          Whether the overlay should be downloaded.
 |      ignore_version : bool
 |          Indicate whether or not to ignore the driver versions.
 |
 |      Note
 |      ----
 |      This class requires a Vivado TCL file to be next to bitstream file
 |      with same name (e.g. `base.bit` and `base.tcl`).
 |
 |  select_pmoda(self)
 |      Select PMODA in the shared pins.
 |
 |      This is done by writing a `0` default) to the `pin_select`
 |      GPIO instance.
 |
 |  select_rpi(self)
 |      Select RASPBERRYPI in the shared pins.
 |
 |      This is done by writing a `1` to the `pin_select`
 |      GPIO instance.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from pynq.overlay.Overlay:
 |
 |  __dir__(self)
 |      __dir__() -> list
 |      default dir() implementation
 |
 |  __getattr__(self, key)
 |      Overload of __getattr__ to return a driver for an IP or
 |      hierarchy. Throws an `RuntimeError` if the overlay is not loaded.
 |
 |  download(self, dtbo=None)
 |      The method to download a full bitstream onto PL.
 |
 |      After the bitstream has been downloaded, the "timestamp" in PL will be
 |      updated. In addition, all the dictionaries on PL will
 |      be reset automatically.
 |
 |      This method will use parameter `dtbo` or `self.dtbo` to configure the
 |      device tree.
 |
 |      Parameters
 |      ----------
 |      dtbo : str
 |          The path of the dtbo file.
 |
 |  is_loaded(self)
 |      This method checks whether a bitstream is loaded.
 |
 |      This method returns true if the loaded PL bitstream is same
 |      as this Overlay's member bitstream.
 |
 |      Returns
 |      -------
 |      bool
 |          True if bitstream is loaded.
 |
 |  load_ip_data(self, ip_name, data)
 |      This method loads the data to the addressable IP.
 |
 |      Calls the method in the super class to load the data. This method can
 |      be used to program the IP. For example, users can use this method to
 |      load the program to the Microblaze processors on PL.
 |
 |      Note
 |      ----
 |      The data is assumed to be in binary format (.bin). The data name will
 |      be stored as a state information in the IP dictionary.
 |
 |      Parameters
 |      ----------
 |      ip_name : str
 |          The name of the addressable IP.
 |      data : str
 |          The absolute path of the data to be loaded.
 |
 |      Returns
 |      -------
 |      None
 |
 |  pr_download(self, partial_region, partial_bit, dtbo=None)
 |      The method to download a partial bitstream onto PL.
 |
 |      In this method, the corresponding parser will only be
 |      added once the `download()` method of the hierarchical block is called.
 |
 |      This method always uses the parameter `dtbo` to configure the device
 |      tree.
 |
 |      Note
 |      ----
 |      There is no check on whether the partial region specified by users
 |      is really partial-reconfigurable. So users have to make sure the
 |      `partial_region` provided is correct.
 |
 |      Parameters
 |      ----------
 |      partial_region : str
 |          The name of the hierarchical block corresponding to the PR region.
 |      partial_bit : str
 |          The name of the partial bitstream.
 |      dtbo : str
 |          The path of the dtbo file.
 |
 |  reset(self)
 |      This function resets all the dictionaries kept in the overlay.
 |
 |      This function should be used with caution. In most cases, only those
 |      dictionaries keeping track of states need to be updated.
 |
 |      Returns
 |      -------
 |      None
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from pynq.bitstream.Bitstream:
 |
 |  insert_dtbo(self, dtbo=None)
 |      Insert dtbo file into the system.
 |
 |      A simple wrapper of the corresponding method in the PL class. If
 |      `dtbo` is None, `self.dtbo` will be used to insert the dtbo
 |      file. In most cases, users should just ignore the parameter
 |      `dtbo`.
 |
 |      Parameters
 |      ----------
 |      dtbo : str
 |          The relative or absolute path to the device tree segment.
 |
 |  remove_dtbo(self)
 |      Remove dtbo file from the system.
 |
 |      A simple wrapper of the corresponding method in the PL class. This is
 |      very useful for partial bitstream downloading, where loading the
 |      new device tree blob will overwritex the existing device tree blob
 |      in the same partial regio.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from pynq.bitstream.Bitstream:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

In [9]:
```
led0 = base.leds[0]
led0.on()
time.sleep(2)
led0.off()
```

In [10]:
```
#Now let's deal with the two RGBLEDs
from pynq.overlays.base import BaseOverlay
import pynq.lib.rgbled as rgbled
import time
base = BaseOverlay("base.bit")
```

In [13]:
```
help(rgbled)
```

```
Help on module pynq.lib.rgbled in pynq.lib:

NAME
    pynq.lib.rgbled

DESCRIPTION
    #  Copyright (c) 2016, Xilinx, Inc.
    #  All rights reserved.
    #
    #  Redistribution and use in source and binary forms, with or without
    #  modification, are permitted provided that the following conditions are met:
    #
    #  1.  Redistributions of source code must retain the above copyright notice,
    #      this list of conditions and the following disclaimer.
    #
    #  2.  Redistributions in binary form must reproduce the above copyright
    #      notice, this list of conditions and the following disclaimer in the
    #      documentation and/or other materials provided with the distribution.
    #
    #  3.  Neither the name of the copyright holder nor the names of its
    #      contributors may be used to endorse or promote products derived from
    #      this software without specific prior written permission.
    #
    #  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
    #  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
    #  THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
    #  PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
    #  CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
    #  EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
    #  PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
    #  OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
    #  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
    #  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
    #  ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CLASSES
    builtins.object
        RGBLED

    class RGBLED(builtins.object)
     |  This class controls the onboard RGB LEDs.
     |
     |  Attributes
     |  ----------
     |  index : int
     |      The index of the RGB LED, from 4 (LD4) to 5 (LD5).
     |  _mmio : MMIO
     |      Shared memory map for the RGBLED GPIO controller.
     |  _rgbleds_val : int
     |      Global value of the RGBLED GPIO pins.
     |
     |  Methods defined here:
     |
     |  __init__(self, index)
     |      Create a new RGB LED object.
     |
     |      Parameters
     |      ----------
     |      index : int
     |          Index of the RGBLED, from 4 (LD4) to 5 (LD5).
     |
     |  off(self)
     |      Turn off a single RGBLED.
     |
     |      Returns
     |      -------
     |      None
     |
     |  on(self, color)
     |      Turn on a single RGB LED with a color value (see color constants).
     |
     |      Parameters
     |      ----------
     |      color : int
     |          Color of RGB specified by a 3-bit RGB integer value.
     |
     |      Returns
     |      -------
     |      None
     |
     |  read(self)
     |      Retrieve the RGBLED state.
     |
     |      Returns
     |      -------
     |      int
     |          The color value stored in the RGBLED.
     |
     |  write(self, color)
     |      Set the RGBLED state according to the input value.
     |
     |      Parameters
     |      ----------
     |      color : int
     |          Color of RGB specified by a 3-bit RGB integer value.
     |
     |      Returns
     |      -------
     |      None
     |
     |  ----------------------------------------------------------------------
     |  Data descriptors defined here:
     |
     |  __dict__
     |      dictionary for instance variables (if defined)
     |
     |  __weakref__
     |      list of weak references to the object (if defined)

DATA
    RGBLEDS_START_INDEX = 4
    RGBLEDS_GPIO_OFFSET = 0
    RGB_BLUE = 1
    RGB_CLEAR = 0
    RGB_CYAN = 3
    RGB_GREEN = 2
    RGB_MAGENTA = 5
    RGB_RED = 4
    RGB_WHITE = 7
    RGB_YELLOW = 6
    __copyright__ = 'Copyright 2016, Xilinx'
    __email__ = "pynq_support@xilinx.com"

AUTHOR
    Graham Schelle

FILE
    /usr/local/lib/python3.6/dist-packages/pynq/lib/rgbled.py
```

In [14]:
```
led4=rgbled.RGBLED(4)
led5=rgbled.RGBLED(5)
```

In [15]:
```
#RGBLEDs take a hex value for color
led4.write(0x7)
led5.write(0x4)
```

In [16]:
```
led4.write(0x0)
led5.write(0x0)
```