Smart Mirror

55°
Light Rain
La Jolla, CA
01:12 pm
Monday
Feb 27, 2017
Today's Headlines
Trump to propose 10 percent spike in defense spending, massive cuts to other agencies
Everything We Know About the Oscars Best Picture Envelope Mix-Up
Trump urges lawmakers to work together to 'save Americans from Obamacare'

Smart Mirror

Originally developed by HackerHouse

Modified by Project-in-a Box developers:
by Luke Wulf and Matthew Kohanfars

Dear Student,

The ECE 196 course and Project in a Box team would like to give credit to HackerHouse for making their smart mirror, which inspired us to create a similar project as one of the choice of this class' projects. We hope, through the process of building this device, you will gain a fundamental understanding of python and the linux operating system.

Sincerely,
P.I.B Team

Description:

This is a project that can be split into two main components: software development and consumer design. The mirror contains a hidden display that runs a python program to display the user's weather, time and news. The student will learn to code this script and develop an elegant display so that this mirror can fit seamlessly into the home environment.

Objective:

This project was developed to expand an electrical engineering student's interdisciplinary value by teaching that student the basics of python and computer engineering. By the end of this project, the student should be comfortable with: writing simple, object-oriented python code, navigating the unix operating system, and designing peripheral parts and housings.

Overview:

Challenge #1: Hello World!

Challenge #2: Python Codecademy

Challenge #3: Software Development and Utility

Challenge #4: Implementing a Graphical User Interface

Challenge #5: Final Assembly and Testing

Challenge #6: Creating the Mirror

Challenge #7: Putting it All Together

Parts List:

1. 1 - Raspberry Pi 3
2. 1 - 16 gb MicroSD card
3. 1 - Laptop Display Screen
4. 1 - Laptop Display Controller
5. 1 - HDMI Cable
6. 1 - 12v Power Supply
7. 1 - 5v Power Supply
8. 1 - USB Keyboard
9. 1 - USB Mouse
10. 1 - 12x18 sheet of mirror film
11. 1 - Set of film application tools
12. 1 - Sheet of Acrylic
13. 2 - Sheets of Plywood
14. 1 - Bottle of super glue or wood glue

Optional Supplies:

1. 1 - Raspberry Pi 3 case (recommended laser cut or 3d printed)
2. 1 - Laptop with SSH compatibility

Challenge #1: Hello World!

The first step of this project is an easy one: designing what may be your first python script. As is tradition with every programming language, you're first task is to write a program that simply outputs "Hello World!" to the console.

Requirement for checkoff: Demonstrate that your HelloWorld.py program works correctly.

Goals of this Challenge #1:

- Set up your Raspberry Pi
- Use the Raspberry Pi to create HelloWorld.py

Setting Up the Raspberry Pi:

- Task: Install an Operating System.
  - Since Raspberry Pi's are very small computers, they will need an operating system in order to function. Also, like computers, the Raspberry Pi will need a display and input devices in order to get anything useful from it.
- Your Job:
  - Use Raspberry Pi's Documentation Page for your initial setup of the Pi.

Use the Raspberry Pi to create HelloWorld.py

- Task: Install Vim:
  - Python is a very easy language to start and learn. However, you need to set up a text editor before you can write a program. There is actually a single command you can insert into your command line to get the popular text editor vim
- Your Job:
  - Insert the command: `sudo apt-get install vim` into your Raspberry Pi's command line.
- Breaking down `sudo apt-get install vim`
  - `sudo` = running with admin permission, you need this to change most things on the Raspberry Pi
  - `apt-get` = argument saying that you wish to get an application from the internet
  - `install` = argument telling the Pi to install vim, other choice can be update if you already have vim installed
  - `vim` = name of the program you wish to get
- Task: Write HelloWorld.py

    - Vim is a very widely used text editor on machines running similar software to the Raspberry Pi. To open vim just type: `vim HelloWorld.py` this creates a new file on your Pi named HelloWorld.py, this is where you will write your code.
- Your Job:
    - Write a program to output “Hello World!” to your command line
    - Find out what this function is by searching through the Python documentation or online. Getting over small hurdles by yourself is the first step into becoming a great programmer!
    - Hint: HelloWorld.py only needs one line of code
    - Hint: To save and close your file in vim, type **`:wq`**
    - Hint: To run your program type `python HelloWorld.py`
    - If you have any issues with vim, consult this cheatsheet

Make sure to get checked off before moving to challenge 2.

Challenge #2: Python Codeacademy

As an alternative sequence, this project has a lot of coding in it. In order to save yourself the headache of debugging simple statements and tripping over syntactic pitfalls it is very important to have a basic understanding of Python. CodeAcademy is a wonderful online and interactive resource that can teach you the basics of Python.

Requirements for Checkoff: Complete Sections 1-9 and 11 in Python Codeacademy

CodeAcademy

Tips for figuring out what your error is:

- Backtrack the lessons in CodeAcademy to see if a concept was taught previously
- Use print statements to output your data to the console
  - ex. `print(%[dataType], data)`
  - Where `[dataType]` could be s, d, f, etc depending on what type of data you are printing
- Tracing your code: Write out on paper what you think is stored in each variable and how they change with each line of code
- Search for the problem online by googling your error message
- Consult a tutor

Please see the page below for important aspects to remember when completing CodeAcademy, and remember to get checked off before moving on to Challenge #3

Big Picture Points of Coding in General:

The MOST important knowledge you can take from CodeAcademy has to do with how you create classes, call methods and assign variables. All 10 lessons will be used in some capacity in the next 2 challenges, but please know this syntax:

Declaring a Variable:

```
[variable name] = data
```

This is one of the simplest lines of code in Python. It sets `[variable name]` to be equal to `data` In this way you can just recall `data` by using `[variable name]`. Also, `[variable name]` can be changed so that it can be used in different ways.

Creating an Object:

```
[variable name] = class(parameters)
```

This line of code creates an object of `class` stored in `[variable name]`.

You must have an object of a class in order to call any of that class' methods. You can think of a class as a schematic and an object as the actual circuit, since you can use the schematic to make multiple of the same circuit, yet when you change one circuit the rest of the circuits do not change. Some classes also take `parameters` as a way to change the "circuit" at initialization.

Calling a Function:

```
[variable name].function(parameters)
```

This line of code executes `function`, that resides within the object named `[variable name]`

Some functions take `parameters` which are just additional pieces of information that makes the same function perform different tasks based on different parameters

Challenge #3: Software Development and Utility

Now you will complete the meat of the project, coding the functionality and utility of the Smart Mirror. Don't be afraid, it should not be too difficult now that you have a newfound knowledge of Python and the online resources of CodeAcademy and the official Python documentation.

Goals of Challenge #3:

- Create the temperature class
- Create the clock class
- Create the news class

Requirement for Checkoff: Show a tutor that your real-time clock, temperature and news modules work correctly by using print statements to the console.

Step 1: Obtain the Starter Code

Task: Obtain the starter code found in the same folder as this documentation. As you can see from the code: we give you the class descriptions, the function headers and descriptions of the general functionality for each class. It is your job to implement this functionality in Python.

Task: Type: `sh ./setup.sh` into the command line. This just downloads a bunch of libraries and packages that you will need in this project.

Step 2: Look at the Pertinent Libraries sheet:

Task: Since there are already pre-built classes that Python allows us to use, it seems wasteful to remake every miniscule class we want to use. At the end of this challenge is a list of some libraries and many resources that you might find useful for your program. To use one of these libraries just call `library.function()` Note: the library names are in the header of your starter code within the import statements, if you need to use a library not contained in this header just write another import statement.

Step 3: Time to Code!

Task: Implement the code left blank in the starter code, when you are done simply type `python SmartMirror.py` into the command line to run your program.

Descriptions of the Classes You Will Implement:

The Clock Class:

- The simplest of the three classes, the clock class has 2 functions
    - __init__

        - the constructor or initializer for the class, in this function you set up all future variables you will need in tick()
    - tick
        - the method that actually obtains and updates the time of the clock
        - Prints the clock time to console
- Remember to consult the Pertinent Libraries sheet if you are having trouble with where to start!

The Weather Class:

- In the weather class, you must first trace your ip address to find your physical latitude and longitude, and then use those two measurements to obtain your local weather.
- The control flow works as follows:
    - Create a weather object
    - Get the ip through that weather object
    - Use that ip to get the local weather of your location.
    - Print the local weather to console
- The weather class has 3 functions:
    - __init__
        - The constructor where you proclaim all future variables you might need (possibly the temperature, local weather, etc).
    - get_ip
        - Obtains the ip address and sets your smart mirror's variables named latitude, longitude and ip. Given as a template for future API requests.
    - get_weather
        - Uses the ip, latitude and longitude to obtain readings of your local weather
- Once again, consult the pertinent libraries sheet for a start position.
- Suggested resources to use:
    - JSON
    - Requests
    - Dark Sky
    - Free Geo IP

The News Class:

- In the news class, you will get the current headlines from the web and display them to your console
- The control flow works as follows:
    - Create a news object
    - Get current headlines by requesting to a news api site.
    - Print the headlines to console
- The news class has 2 functions:
    - __init__:

        - The constructor that stores class variables such as the strings of the headlines
    - get_headlines:
        - Requests to a news api and then parses through the output to obtain the current day's headlines
- Once again, consult the pertinent libraries sheet for help
- Suggested Resources
    - JSON
    - Requests
    - NewsAPI.org

Global Variables:

- Frequently in coding, you may want to have variables that are accessible to all of the classes in your program. To achieve this we use a thing called global variables.
- Global Variables are declared before any of your classes and then can be used inside your class' methods.
- Some examples of good global variables for this project are:
    - IP Address
    - API Tokens
    - Any other API variables
    - Think of more!
- You can declare your global variables using variable declarations below the comment:

```
#-------------------Global-Variables-----------------------------
```

Pertinent Libraries and Definitions:

Libraries:

- TKinter
    - strftime
    - Requests
    - JSON

Definitions:

- Tokens: Unique ID's that you need in order to use an API. It tells the API who you are
- JSON: Class that decodes a url's outputted text into a usable Python dictionary
- Requests: Used to ping a website and request data from said site.

API's:

- API
    - DarkSky
    - FreeGeoIP
    - NewsAPI

- Feel free to experiment with what API's you use! There are a ton of them online that can all do various functions in relatively the same JSON format.
- For API's: You will need to sign up and get your own unique API key. Use this key in your requests to get information. Most API's are free!

Challenge #4: Implementing a Graphical User Interface

Now that you have your weather, clock and news classes working, you are now going to display this data in an elegant matter on your screen. This display is called a GUI or Graphical User Interface, and it is what you, the user, will see when you run your program. Code output, by itself, is usually a very ugly thing since it is mostly limited to print statements, however if you wanted to make an app, videogame or webpage you should know how to create a GUI.

Goals for Challenge #4:

- Format your clock, weather and news data in a readable manner on your screen by creating a Smart Mirror GUI

Requirement for Checkoff: Show a tutor that your Smart Mirror is displaying your data in an elegant, non-command-line, format.

GUI Components: The GUI that you need to create does not need to have any animations or moving parts. Actually is is just a segmented screen where certain areas of the screen display different information. The most simple of GUIs are made from many frames and text boxes.

- A **Window (TK)** is the largest GUI object, it is the all encompassing pop-up where your GUI resides in.
- A **Frame** is an invisible box that keeps all of the GUI objects within it constrained to this invisible box. Frames can be placed in the nine main locations: top, bottom, left, right, middle, top-left, top-right, bottom-left, and bottom-right. You can also place frames within frames to segment your window space even further
- A **Text Box** is a GUI object that you place inside of a frame to display text within your GUI.

So, the general flow of building a GUI is:

1. Make a window `variable = Tk()`
2. Put frames in the window corresponding to the areas you want your data to be displayed in.
3. Put text boxes containing your data within these frames to display your data

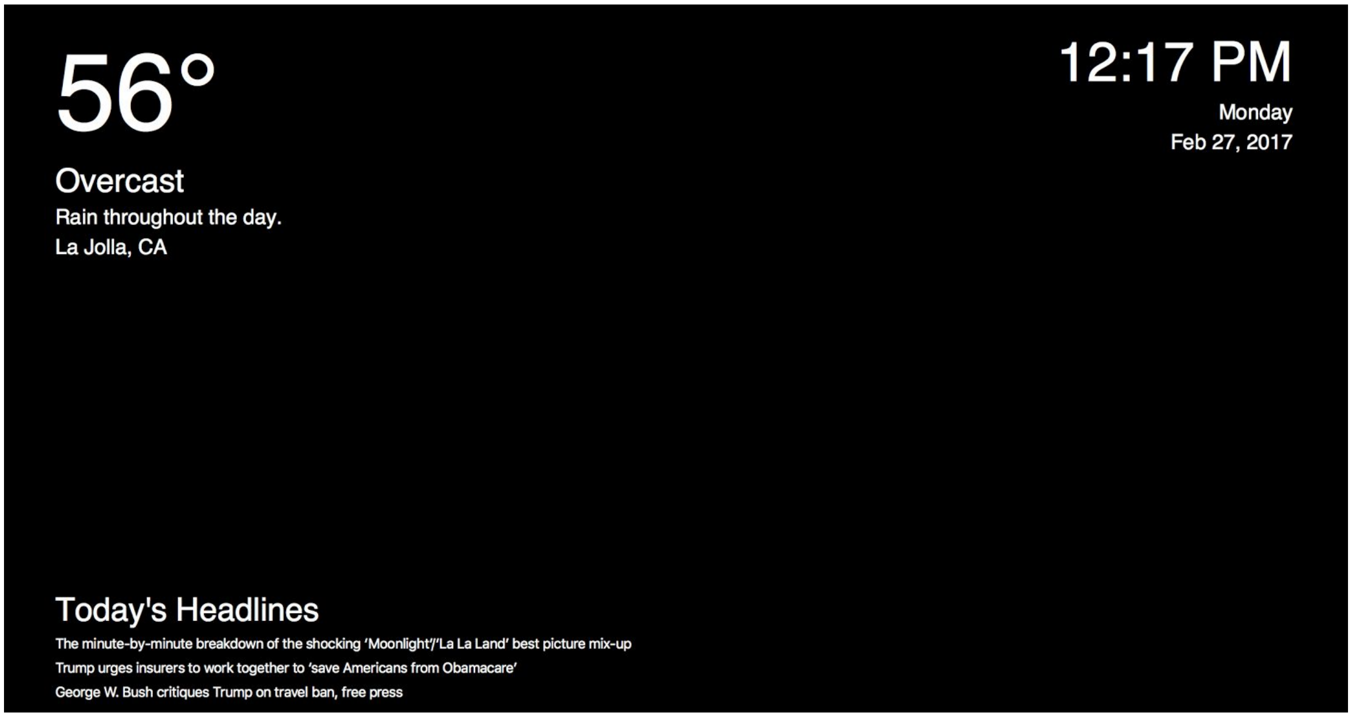
Useful Links:

- Tk() is synonymous with window
- Frame
- Text Box
- .pack() :: Can be used in any TKinter GUI object. It locks the object to a position

Useful Notes:

- You should append the code you wrote for the console output to instead display your data within a frame, then when you call the tick(), get_weather(), and get_headlines() methods you can simply give that data to your window and display it all at once.

- The best color scheme to see your data through the mirror is a black background with white text
- You will need to change all your classes to extend from Frame to work with the GUI, to do this replace all class headers with `Class(Frame)` instead of `Class()`
- Since you are now extending from Frame you should change all your `__init__(self)` functions to `__init__(self, parent, *args, **kwargs)` The extra parameters are used for the Frame class.
- Uncomment the DisplayWindow class at the bottom of your file for a start point in making your GUI.
- Here's an example of the output



56°

Overcast

Rain throughout the day.

La Jolla, CA

12:17 PM

Monday

Feb 27, 2017

Today's Headlines

The minute-by-minute breakdown of the shocking 'Moonlight'/'La La Land' best picture mix-up

Trump urges insurers to work together to 'save Americans from Obamacare'

George W. Bush critiques Trump on travel ban, free press

Challenge #5: Final Assembly and Testing

Making the Display Vertical:

To make the display vertical, simply open up config.txt which is located in your Raspberry Pi's `boot` folder.
(make sure you are opening the file in the boot folder and not creating a new file)

```
sudo vim /boot/config.txt
```

Add line at bottom of config.txt called

```
display_rotate = 1
```

Then, simply save and quit from vim using

```
:wq!
```

Once you restart your Pi, you should now be in portrait mode.

Challenge #6: Creating the Mirror

Now that all the code aspect is completed it's time to build the framework for your mirror!

Reflective Surface
Items Needed:

- Sheet of Mirror Film
- Solution Spray (water may be used)
- Squeegee
- Cloth
- Blade
- Acrylic sheet (The figure uses a ⅛ in piece of acrylic)

Using a piece of acrylic (12x18), use the blade to cut the filament to the size of the acrylic sheet. Before applying the filament, clean off the surface of the acrylic then apply the solution and place the filament. Use the squeegee and navigate over the filament and take out any bubbles located on the surface.

Note: Placing the cloth over the squeegee and then navigating across the sheet will prevent scratches on the reflective filament.

Inside the Mirror
In this segment, 2 pieces of plywood were used. A ¼in piece of plywood to hold the mirror in place and ⅛in piece of plywood used as the supporting sheet in the back of the mirror. In the figure to the right, the outside border was cut to size of the monitor to hold it into place and stained. On the left figure, the sheet was cut and holes were drilled to fit the wires, monitor circuit board, and Raspberry Pi.

Enclosures for Components
To clean up the backside of the mirror create enclosures for the different components. A 3-D printer was used for the enclosure of the Raspberry Pi and an acrylic enclosure was created for the monitor circuit board.

The Frame
The frame must contain a width that can fit the interior pieces that were created before.

Challenge #7: Putting it All Together

After the reflective surface, plywood pieces, enclosures and frame are created we are now ready to put everything together!

Resources

This project was inspired by: HackerHouse

Raspberry Pi: https://www.raspberrypi.org/

Code Academy: https://www.codecademy.com/learn/python

TutorialsPoint: https://www.tutorialspoint.com/python/