

# LAB 2: CODING WITH HTML

This lab is the second in a series. In this lab the student will learn about the features of the HTML coding language. Students will also be exposed to their first instance of CSS. The ultimate objective of this lab is to set up the Raspberry Pi so that the student can work towards completing the RetroPie Video Game Emulation Labs.

IT 4530 Senior  
Capstone Project  
Spring 2019  
Georgia Southern  
University

# TABLE OF CONTENTS

## Contents

INTRODUCTION TO HTML.....	3
File Setup.....	3
Saving Your File .....	3
<!DOCTYPE HTML> .....	5
<HTML> Tag .....	5
<Head>.....	5
<Title> .....	6
<Body> .....	6
Anything there?.....	7
<header>.....	8
<p> .....	8
  .....	9
Unordered Lists .....	10
Ordered Lists .....	12
Textbox .....	13
Buttons .....	13
Class.....	15
ID .....	16
Style .....	16
<a> .....	17
<hr> .....	17
<footer> .....	17
Install Scrot .....	19
Screenshot .....	19
Instant.....	19
Delayed .....	19

**IMPORTANT:** Make sure that throughout the lab you work in **one** Html document. This file will be the building block for future labs.

# INTRODUCTION TO HTML

HTML, Hypertext Markup Language, is the main language used to code and display websites. Built with basic tags, html is versatile and easy to use in any text editor.

## FILE SETUP

### Saving Your File

For coding HTML files, we'll be using IDLE. On your Raspberry Pi, IDLE is a python editor but we can use it as a basic text editor.

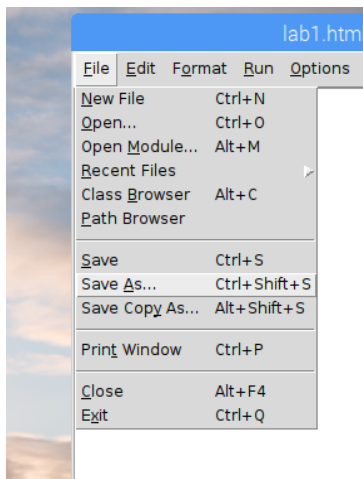
When you open IDLE make sure to select File, and new. This will open a window for you to type and save a file.

In the upper left corner click on file (Figure 1). Click on Save As and navigate in the file explorer to where you would like to save your file. Please save your file as lab1.html (Figure 2). Please ensure the .html is at the end of your file as it is important for the web browser to be able to run your file.

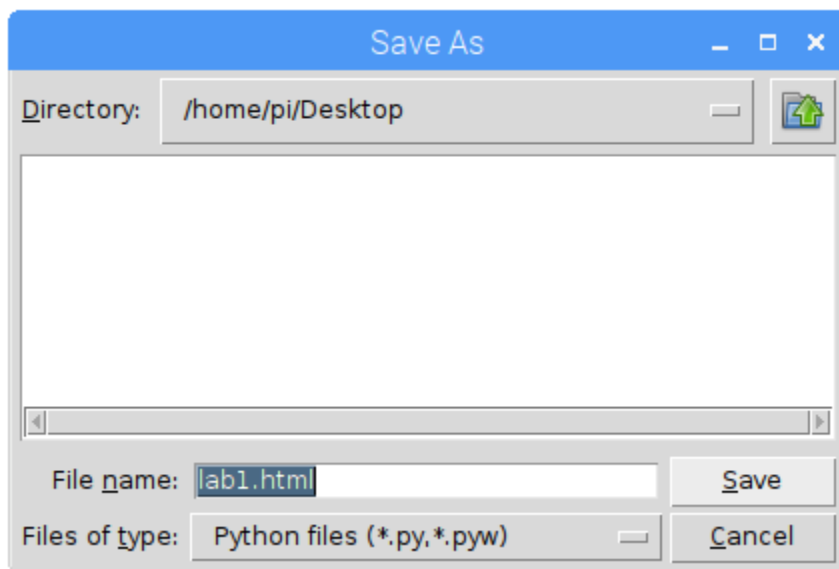
All html files should have a .html extension.

**It is important to remember the mantra *Save Early, Save Often* as you complete your assignments as computers and programs like to crash unexpectedly.**

**Figure 1**



**Figure 2**

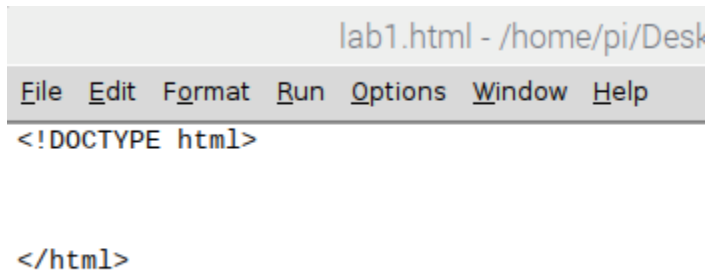


## <!DOCTYPE HTML>

All HTML documents are required to have a declaration at the top of the page. The declaration tells the web browser what type of it should expect to render. In HTML, the declaration is <!DOCTYPE HTML>. This declaration is always inserted at the top of the page. If you fail to include this declaration or do not place it properly the file will not load.

As you can see in Figure 3 we have placed the doctype declaration at the top of the file.

**Figure 3**



```
lab1.html - /home/pi/Desktop
File Edit Format Run Options Window Help
<!DOCTYPE html>

</html>
```

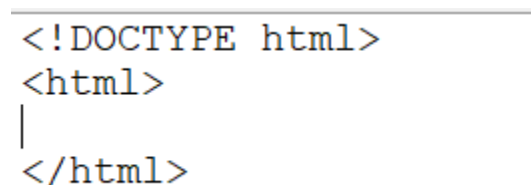
## <HTML> Tag

In any HTML document you need two HTML tags. One tag will open, the <HTML> tag. One tag will close, the </HTML> tag. You place the opening tag <HTML> (Figure 4) immediately after the doctype declaration tag. You place the closing tag </html> immediately after the opening tag. Any code that is HTML will be placed inside of these two tags.

**If you do not place the code within these two tags that code will not run.**

Notice how the closing tag has a forward slash. This standard will apply to most <HTML> features with the exception of a few. We will notify you when you do not need to include a closing tag.

**Figure 4: Opening and Closing Html**



```
<!DOCTYPE html>
<html>
|
</html>
```

## <Head>

Now it's time to divide the html document into two parts, The Head and Body. The head tags always comes first and help define the document. This tag will contain the metadata for a document such as, the website title, the style, and any links to other documents.

Once again create an opening and closing tag for the <head>, placed within the <html> tags. (Figure 5)

**Figure 5**

```
<!DOCTYPE html>
<html>
<head>
```

```
</head>
```

### <Title>

The Title Tag includes what will be shown on the web page tab, as such it is the title for your page, so have fun with it, but try to keep it within the scope of the page. the <title> tags go within the <head> tags and can be written in the same line or on separate ones. Whichever you prefer. (Figure 6)

**Figure 6**

```
<!DOCTYPE html>
<html>
<head>

    <title>My Web Page</title>

</head>
```

---

### <Body>

The second part of an html document is the Body. The body contains all of the visual content within the document. You can include text, pictures, forms, links, lists, tables and much more.

The <body> tags will be placed within the <html> tags but outside of the <head>. See Figure 7.

**Figure 7**

```
<!DOCTYPE html>
<html>
<head>

    <title>My Web Page</title>

</head>
<body>

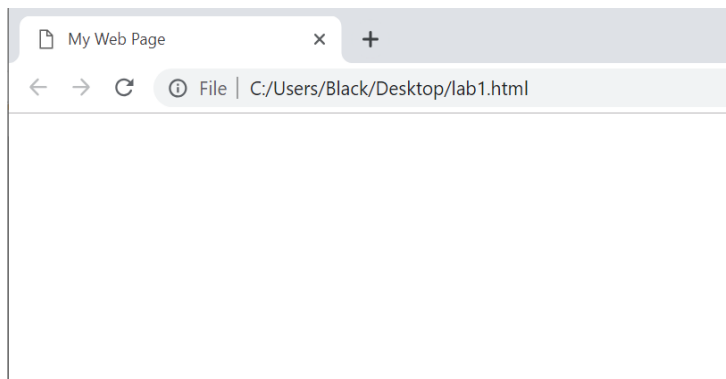
|
</body>
</html>
```

If you like more white space between tags you can add lines if you want. Lines without tags are not read in the final file, adding these can help you read the file better and help troubleshoot coding errors by keeping it neat.

## Anything there?

Now we can check and see what the page looks like so far. by double clicking the icon of our html file on the desktop, we can see it displayed in a browser. This should open a browser window, display a blank page and your title on the tab or title bar. See Figure 8.

**Figure 8**



As you can see, the page is blank so why don't we fill it?



# INPUT INFORMATION

Html files can be populated with a number of different tags to input an array of information and display it to the viewer. This is done by placing the tags within the <body> tags.

**This next section will work exclusively in the <body>.**

## <header>

The header tag is used to label sections within a website. Items within the header tag appear at the top of the page just like a word document and can be filled with different sized text with the use of <h1> tags. <h> tags scale is size, <h1> being the largest font, and <h6> being the smallest.

Let's give our page a header, remember the opening and closing tags, and within the <header> tags we will use <h1> for a large bold title. See Figure 9 if you get confused and to check your work.

**Figure 9**

```
<body>

    <header><h1 style="text-align:center">Pi Labs - Html</h1></header>

</body>
```

Once again , save and run your page in your preferred browser to see the changes.

## <p>

Called the Paragraph tag, it is used to type blocks of text within the html file. Think of it just like writing within a word document, this tag will make up the paragraphs you want to show to your viewers. <p> tags still need an opening tag and a closing tag like always, and when put into the <body>, the html automatically gives the block its own margins. These margins can be modified later with CSS, so don't worry too much about them for now.

Let's add some flavor text, be creative.

**Figure 10**

```
<body>

    <header><h1 style="text-align:center">Pi Labs - Html</h1></header>
    <p>Html is fun and easy to use!</p>
    |
</body>
```

---

## <br>

The page is looking good so far, but maybe it's a bit empty? why not add some more <h1> and <p> tags to fill in some space. Adding in more would be fine but we'd much rather have it look nice. This is where the <br> tag comes into play. The <br> tag creates a line of blank space within the file, like empty lines in the code but they show up in the page. This creates a neat and polished look to your page.

Figure 11 and 12 will show the difference between not using <br> and using them.

Figure 11: Without <br>

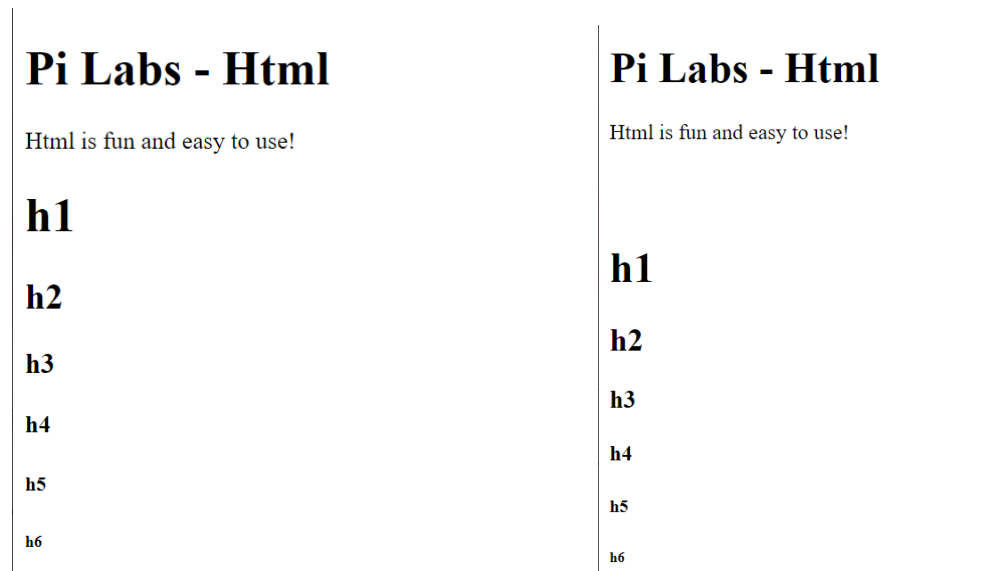
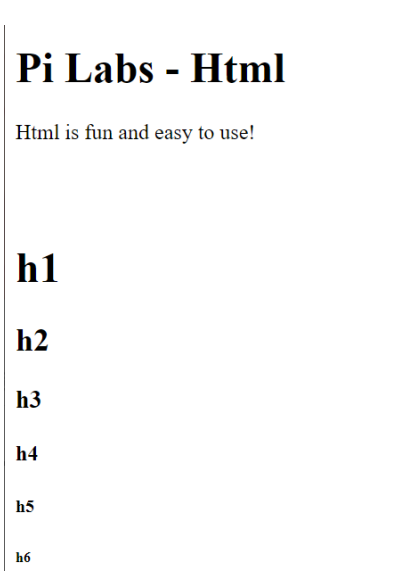


Figure 12: With 2 <br>



Reminder: <br> tags are some of the few tags that don't require a closing tag.

# CREATE LISTS

Lists are a great way to organize data into an easily readable format. In Html we can create two types of lists: Unordered and Ordered. We will cover both in this section, and once again continue placing these tags within the <body> of the document.

## Unordered Lists

Unordered lists show up as a bulleted list on an html page, and are simple to make. All lists require two types of tags. The first tag states the type of list, and the second is for holding each item. For an unordered list, we start with <ul> and the closing </ul>. Then each element of the list goes between these two with the tags <li> and </li>.

**Figure 13**

<body>

```
<header><h1>Pi Labs - Html</h1></header>
  <p>Html is fun and easy to use!</p>
  <br>
  <br>
  <h1>h1</h1>
  <h2>h2</h2>
  <h3>h3</h3>
  <h4>h4</h4>
  <h5>h5</h5>
  <h6>h6</h6>
  <br>
  <h3>Unordered List</h3>
    <ul>
      <li>List Item</li>
      <li>List Item</li>
      <li>List Item</li>
      <li>List item</li>
    </ul>
```

As you can see in Figure 13, we've added a header for our list, this way we can describe what's going to go inside of it. Between each `<li></li>` we can input anything, why don't we do a list of foods that you like? We can even change the header to something appropriate. See Figure 14 for how your html should look when run in chrome.

**Figure 14**

# Pi Labs - Html

Html is fun and easy to use!

**h1**

**h2**

**h3**

**h4**

**h5**

**h6**

## **Favorite Foods**

- Mac and Cheese
  - Pizza
  - Tacos
  - Sushi
-

## Ordered Lists

Ordered lists have the same general format as their bulleted counterpart except these lists are numbered. Ordered Lists are good for rankings, writing steps or instructions, and much more. The main tag for an ordered list is `<ol></ol>`, with the same `<li></li>` tags for the items within the list. Using an ordered list, put in a few of your favorite video games in order.

**Use a `<br>` between the two lists to keep things from running into each other.**

**Figure 15**

```
<h3>Favorite Foods</h3>
    <ul>
        <li>Mac and Cheese</li>
        <li>Pizza</li>
        <li>Tacos</li>
        <li>Sushi</li>
    </ul>
<br>
<h3>Favorite Games</h3>
    <ol>
        <li>Dark Souls</li>
        <li>Kingdom Hearts 3</li>
        <li>Breath of the Wild</li>
        <li>Bloodborne</li>
    </ol>
<br>
```

---

# CREATING INPUT

Most Websites include places for users to interact or input their own data. Textboxes and Buttons make up the general majority of these interactive elements. Both of these elements can be made using the `<input>` tag.

**Input tags don't include a closing tag.**

The input tag can be used multiple ways and has many modifiers, main examples include : Type , Name, and Value. Using these extra attributes allows the html file to keep track of what it needs to display.

Type : Specifies the type of element to display. Examples : textbox, submit, button, image.

Name : Specifies the name of the input, this helps for formatting and linking.

Values : Specifies the text that is displayed within the element. Example : What is shown on a button.

## Textbox

let's make a simple textbox and give it a header as well. Once again using `<h3>` to keep it a reasonable size. After which we need to declare the input : `<input type="textbox">`. After adding this, the page should show a small textbox.

**Figure 16**

```
<br>
<h3>Textbox</h3>
      <input type="textbox">
<br>
```

## Buttons

Buttons are common elements in web pages, Submit and Search are two very common ones. Once we can use the `<input>` tag to create a button simply by changing the type.

```
<input type="submit">
```

But a blank button is boring. We can add text to its face by adding in value.

```
<input type="submit" value="Click Me!">
```

Afterwards, we should add a name to the button. This will help specify it in our css file in the subsequent lab.

**Figure 17**

```
<br>
<h3>Textbox</h3>
    <input type="textbox">
<br>
<h3>Submit Button</h3>
    <input type="submit" name="lab1" value="Click Me!">
<br>
```

# CREATE A LINK

This section will cover many finishing touches for a typical html file. Many of these elements are hidden from the main webpage and will be used to connect the html to its css file.

## Class

Class is a functional additions to elements that define what style or section it will be connected to in the CSS file.

Class is an attribute added to an element which applies a certain style to any element that has the same class name. It is an easy way to group headings for specific formats; like making them all bold and changing the colour to red. It's easier than typing an inline style for each element.

Simply add `class=""` into the opening tag of the element you'd like to mark for the CSS file. Remember to keep the names simple and related to what you're formatting

**Figure 18**

```
<h3 class="Li">Favorite Foods</h3>
    <ul>
        <li>Mac and Cheese</li>
        <li>Pizza</li>
        <li>Tacos</li>
        <li>Sushi</li>
    </ul>
<br>
<h3 class="Li">Favorite Games</h3>
    <ol>
        <li>Dark Souls</li>
        <li>Kingdom Hearts 3</li>
        <li>Breath of the Wild</li>
        <li>Bloodborne</li>
    </ol>
<br>
```

---

We chose to use the class "Li" for list, keeping things simple will help troubleshooting problems later on.



## ID

Id is another attribute to elements that helps link elements to functions and css styles. While class covers many different elements, Id is unique to each element. By adding Ids we can make functions move smoothly and diagnose problems that may arise in the code.

Let's add an Id to our textbox and the button.

**Figure 19**

```
<br>
<h3 id="textB">Textbox</h3>
    <input type="textbox">
<br>
<h3 id="Sub">Submit Button</h3>
    <input type="submit" name="lab1" value="Click Me!">
<br>
<br>
```

---

## Style

Style can be used both globally throughout the document or inline for a single element. With style we can modify the look of headers and paragraphs using basic language.

Styles include:

- background-color *for the page background.*
- color *for text colors.*
- font-family *for changing the text font*
- font-size *for changing font size*
- text-align *for aligning text along the page*

We're going to add a text-align style to the First header of our page. Feel free to be creative and try out different styles.

**Figure 20**

```
<body>

    <header><h1 style="text-align:center">Pi Labs - Html</h1></header>
        <p>Html is fun and easy to use!</p>
        <br>
        <br>
```

---

By aligning the first header, it makes our page flow better to the reader.

## <a>

The <a> tag is a common tag that creates a hyperlink on your webpage. There are a number of attributes within the <a> tag but the most important one is href. This is where you will include the link you'd like to go to.

We're going to set up a link to W3Schools, a really helpful website for html, php, and css tutorials.

**<a href="https://www.w3schools.com/html/default.asp">This is a link to the W3Schools website, click here to learn more about html!</a>**

## <hr>

Another good tag for organizing is the <hr> tag. Putting it into your html file like a <br> tag, will create a single horizontal line across the page. This is helpful for sectioning off the page for different groups of data.

We're going to use the <hr> tag at the bottom of our document to include a <footer>. This way we can separate it from the rest of the web page.

### **Figure 21**

```
<br>
<br>
<a href="https://www.w3schools.cc
<hr>
```

---

## <footer>

A <footer> tag in html runs like a footer in Microsoft Word, by reserving a bottom section of a page and contains data like: Your name, date, project or class. You can also use the <footer> tag for copyright data, or a contact section.

In this lab, we will use the <footer> tag to sign our web page. This should include:

- The Lab Number
- The year
- The class
- Your name

Feel free to style this section appropriately. We've decided to align this footer in the center, as well as include a copyright symbol.

**Remember to Include both an open and closing <footer>tag.**

**Figure 22**

```
<hr>

<footer style="text-align:center">
    Lab 1, Spring 2019 Capstone Project
    <br>
    &copy; 2019 Frantom, Fulcher, Guy, White
</footer>
```

---

**Figure 23**

[lick here to learn more about html!](#)

---

Lab 1, Spring 2019 Capstone Project  
© 2019 Frantom, Fulcher, Guy, White

---

# TAKE A SCREENSHOT

This is the end on the html lab and all that is left to do is take a screenshot of your working web page. Taking a screenshot in the Raspberry Pi is easy and only requires a few lines of code to get it working.

**Reminder: Make sure that your Raspberry Pi is connected to the internet through the ethernet port.**

## Install Scrot

Scrot is an easy to use capture application that can be installed through the Raspberry Pi's command terminal.

Open the Terminal with the icon that looks like a computer monitor. Afterwards we will type the following command to get Scrot : **sudo apt-get install scrot**

The application will automatically install and should return a line reading : scrot is installed.

**If this did not happen , please check the spaces and dashes within your line and run it again.**

**See Figure 24 for what the terminal will look like.**

## Screenshot

Now that Scrot is installed you can take two different types of screenshots, Instant and Delayed. An Instant screenshot will include the terminal window while Delayed and be given a set time to run the code to take the picture.

### Instant

To take a screenshot, open the terminal and type : **scrot**

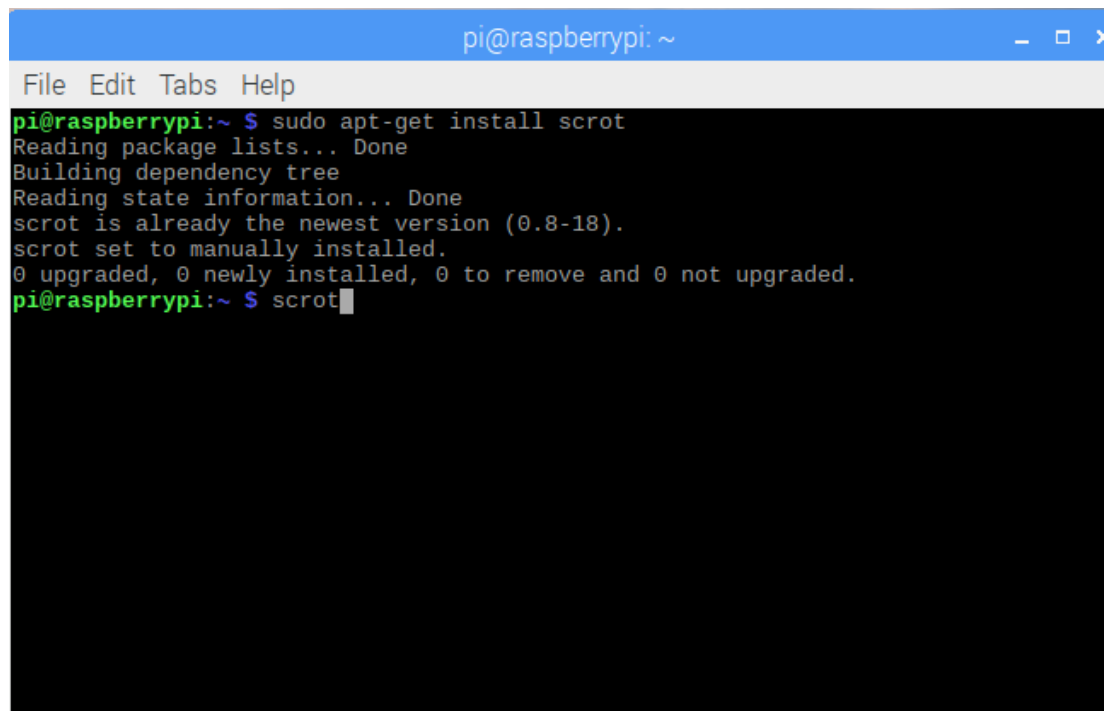
The image should then appear in the /home/pi folder in the Raspberry Pi's File Manager.

### Delayed

To take a screenshot, open the terminal and type : **scrot -d 10**

The 10 in this case indicates the number of second to wait before the picture is taken. You should give yourself enough time to minimize the terminal or switch to viewing your completed website.

**Figure 24**



A terminal window titled "pi@raspberrypi: ~" with a blue header bar. The window contains a menu bar with "File", "Edit", "Tabs", and "Help". The terminal output shows the command "sudo apt-get install scrot" being executed. The output indicates that the package lists are read, the dependency tree is built, and the state information is read. It then states that "scrot" is already the newest version (0.8-18) and is set to manually installed. Finally, it reports that 0 packages were upgraded, 0 were newly installed, 0 were to be removed, and 0 were not upgraded. The prompt "pi@raspberrypi:~" is shown again with the command "scrot" entered and a cursor at the end.

```
pi@raspberrypi:~ $ sudo apt-get install scrot
Reading package lists... Done
Building dependency tree
Reading state information... Done
scrot is already the newest version (0.8-18).
scrot set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ scrot
```