

4.2.2 Raspberry Pi Programming

"Hello world" is the beginning of everything when it comes to computing and programming. It's the first thing you learn in a new programming language, and it's the way you test something out or check to see if something's working because it's usually the simplest way of testing simple functionality.

Warriors of programming language wars often cite their own language's "hello world" against that of another, saying theirs is *shorter* or *more concise* or *more explicit* or something. Having a nice simple readable "hello world" program makes for a good intro for beginners learning your language, library, framework, or tool.

I thought it would be cool to create a list of as many different "hello world" programs as possible that can be run on the Raspberry Pi using its Raspbian operating system, but without installing any additional software than what comes bundled when you download it from the Raspberry Pi website. I've created a GitHub repository of these programs, and I've explained 10 of them for you here.

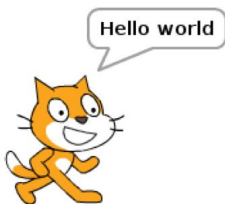
1. Scratch

Scratch is a graphical block-based programming environment designed for kids to learn programming skills without having to type or learn the syntax of a programming language. The "hello world" for Scratch is simple—www.EnggTutorials.com

1. Open **Scratch 2** from the main menu
2. Click **Looks**.
3. Drag a **say Hello!** block into the workspace on the right.
4. Change the text to **Hello world**.



5. Click on the block to run the code.



2. Python

Python is a powerful and professional language that's also great for beginners— and it's lots of fun to learn. Because one of Python's main objectives was to be readable and stick to simple English, its "hello world" program is as simple as possible.

1. Open **Thonny Python IDE** from the main menu.
2. Enter the following code:

```
print("Hello world" )
```

3. Save the file as **hello3.py**.
4. Click the **Run** button.



opensource.co

3. Ruby/Sonic Pi

Ruby is another powerful language that's friendly for beginners. Sonic Pi, the live coding musicsynth, is built on top of Ruby, so what users actually type is a form of Ruby.

1. Open **Sonic Pi** from the main menu.
2. Enter the following code:

```
puts "Hello world"
```

3. Press **Run**.



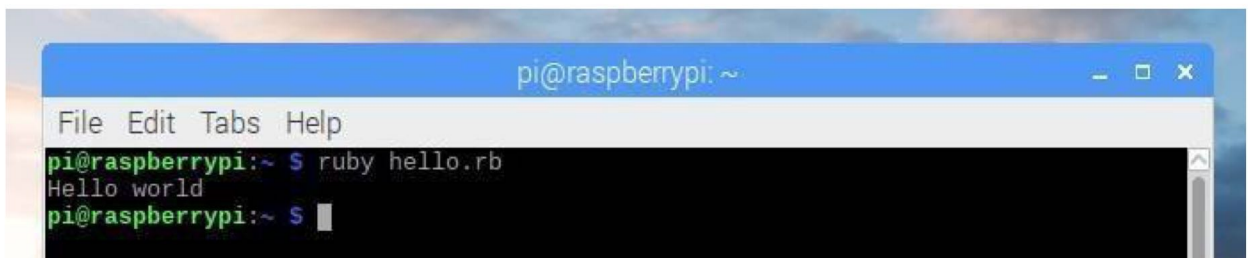
Unfortunately, "hello world" does not do Sonic Pi justice in the slightest

Alternatively, to using the Sonic Pi application for this example, you can write Ruby code in a text editor and run it in the terminal:

1. Open **Text Editor** from the main menu.
2. Enter the following code:
3. Save the file as `hello.rb` in the home directory
4. Open **Terminal** from the main menu.
5. Run the following command:

```
puts "Hello world"
```

```
ruby hello.r
```



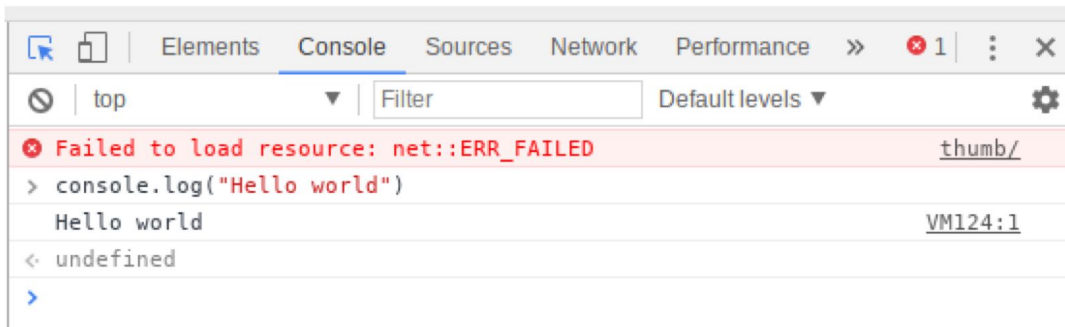
4. JavaScript

This is a bit of a cheat as I just make use of client-side JavaScript within the web browser using the Web Inspector console, but it still counts!

1. Open **Chromium Web Browser** from the main menu.
2. Right-click the empty web page and select **Inspect** from the context menu.
3. Click the **Console** tab.
4. Enter the following code:

```
console.log("Hello world" )
```

5. Press **Enter** to run.



You can also install NodeJS on the Raspberry Pi, and write server-side JavaScript, but that's not available in the standard Raspbian image.

5. Bash

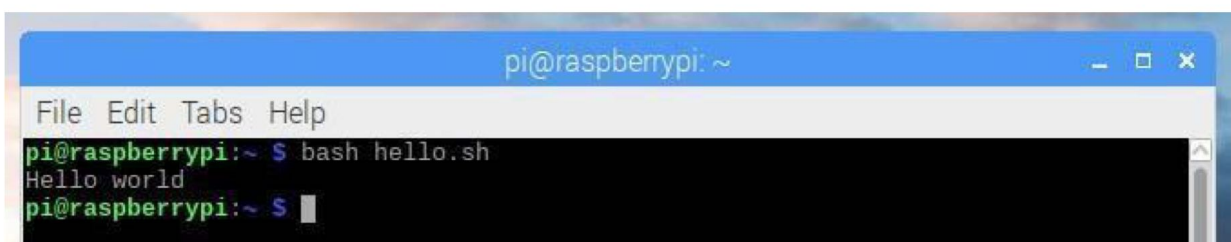
Bash (Bourne Again Shell) is the default Unix shell command language in most Linux distributions, including Raspbian. You can enter Bash commands directly into a terminal window, or script them into a file and execute the file like a programming script.

1. Open **Text Editor** from the main menu
2. Enter the following code:

```
echo "Hello world"
```

3. Save the file as `hello.sh` in the home directory.
4. Open **Terminal** from the main menu.
5. Run the following command:

```
ash
```



Note you'd usually see a "hashbang" at the top of the script (`#!/bin/bash`), but because I'm calling this script directly using the `bash` command, it's not necessary (and I'm trying to keep all these examples as short as possible).

You'd also usually make the file executable with `chmod +x`, but again, this is not necessary as I'm executing with `bash`.

6. Java

Java is a popular language in industry, and is commonly taught to undergraduates studying computer science. I learned it at university and have tried to avoid touching it since then. Apparently, now I do (very small amounts of) it for fun...

1. Open **Text Editor** from the main menu.
2. Enter the following code:

```
3 public class  
  Hello {
```

```
4 public static void main(String[]  
  args) {
```

```
5 System.out.println("Hello world");  
  .
```

```
6 }  
  .
```

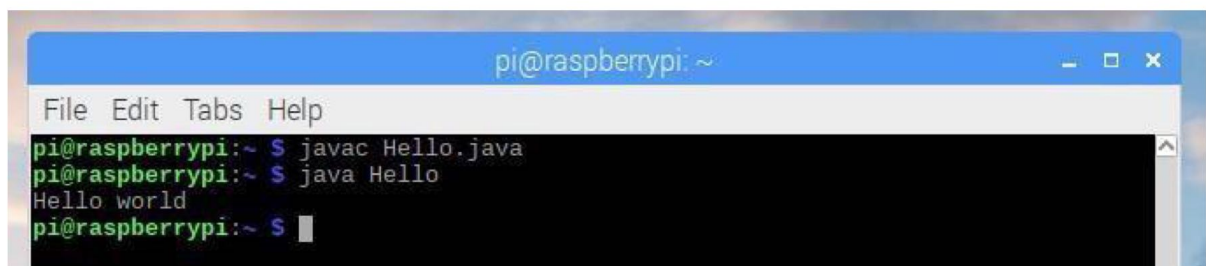
```
7. }
```

```
8.
```

9. Save the file as **Hello.java** in the home directory.
10. Open **Terminal** from the main menu.
11. Run the following commands:

```
12. javac Hello.java
```

```
java Hell o
```



I could *almost* remember the "hello world" for Java off the top of my head, but not quite.

I always forget where the `String[] args` bit goes, but it's obvious when you think about it...

7. C

C is a fundamental low-level programming language. It's what many programming languages are rewritten in. It's what operating systems are written in. See for yourself—take a look at the source for Python and the Linux kernel. If that looks a bit hazy, get started with "hello world":

1. Open **Text Editor** from the main menu.
2. Enter the following code:
3. `#include <stdio.h>`
4. `int main() {`
5. `printf("Hello world\n");`
6. `}`
6. Save the file as `hello.c` in the home directory.
7. Open **Terminal** from the main menu
8. Run the following commands:

```
9 gcc -o hello  
. hello.c
```

```
./hel o
```



Note that in the previous examples, only one command was required to run the code (e.g., `python3 hello.py` or `ruby hello.rb`) because these languages are interpreted rather than compiled. (Actually Python is compiled at runtime but that's a minor detail.) C code is compiled into byte code and the byte code is executed.

If you're interested in learning C, the Raspberry Pi Foundation publishes a book *Learning to code with C* written by one of its engineers. You can buy it in print or download for free.

8. C++

C's younger bother, C++ (that's C incremented by one...) is another fundamental low-level language, with more advanced language features included, such as classes. It's popular in a

range of uses, including game development, and chunks of your operating system will be written in C++ too.

Open **Text Editor** from the main menu. Enter the following code:

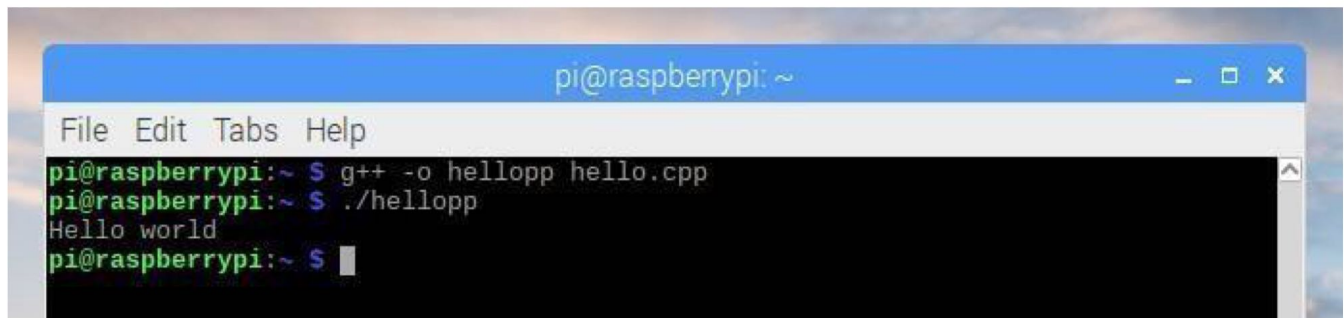
```
#include <iostream> using namespace std; int main() {  
    cout << "Hello world\n";  
}
```

Save the file as **hello.cpp** in the home directory. Open **Terminal** from the main menu.

Run the following commands:

```
1  g++ -o hellopp  
   hello.cpp
```

```
./hellopp
```



The screenshot shows a terminal window titled 'pi@raspberrypi: ~'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal output shows the following commands and results:

```
pi@raspberrypi:~ $ g++ -o hellopp hello.cpp  
pi@raspberrypi:~ $ ./hellopp  
Hello world  
pi@raspberrypi:~ $
```

Readers familiar with C/C++ will notice I have not included the main function return values in my examples. This is intentional as to remove boilerplate, which is not strictly necessary.