

Programming the Raspberry Pi

Jon Flanders
@jonflanders



This module is about..

```
#include <conio.h>
#include <math.h>
#include <dos.h>
#include <stdio.h>

int ok;
int i,j,k;
int ch;
long cod;

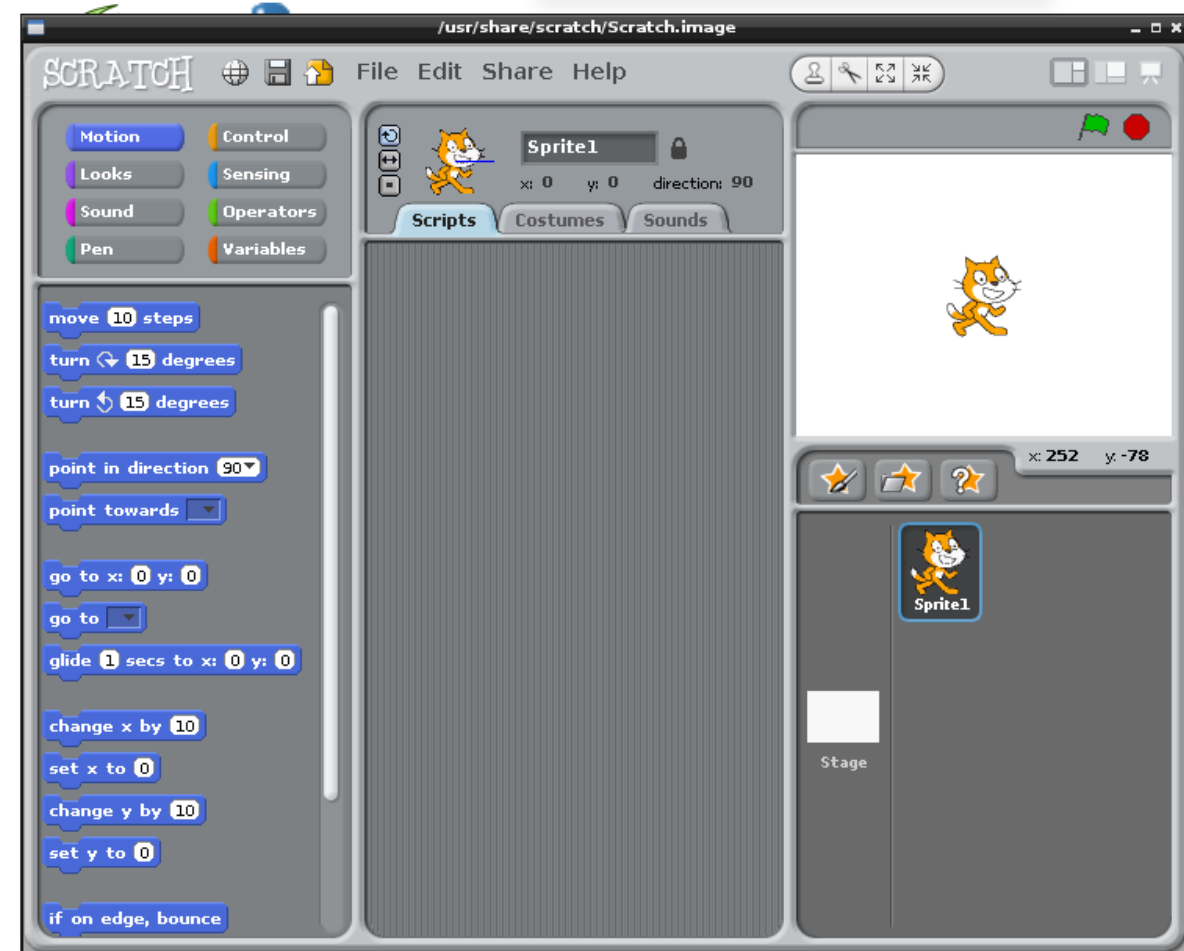
const long F0=32308237;

void mysound(void)
{
    outp(0x42,(cod&255));
    outp(0x42,(cod>>8));
    outp(0x61,(inp(0x61)|3));
    for(i=1;i<=5;i++) {
        delay(10);
        if (kbhit()!=0) break;
    };
    outp(0x61,(inp(0x61)&(~2)));
    ch=getch();
};

int main(void)
{
    ch=getch();
    outp(0x43,0xb6);
    do {
        switch(ch) {
            case '1':cod=F0/120.8;break;
            case '2':cod=F0/136.8;break;
            case '3':cod=F0/164.8;break;
            case '4':cod=F0/184.6;break;
            case '5':cod=F0/196.0;break;
            case '6':cod=F0/210.0;break;
            case '7':cod=F0/220.0;break;
            case '8':cod=F0/234.5;break;
            case '9':cod=F0/247.0;break;
            case '0':cod=F0/261.6;break;
            case 'A':cod=F0/277.0;break;
            case 'B':cod=F0/293.6;break;
            case 'C':cod=F0/311.2;break;
            case 'D':cod=F0/330.0;break;
            case 'E':cod=F0/349.9;break;
            case 'F':cod=F0/370.0;break;
            case 'G':cod=F0/391.9;break;
            case 'H':cod=F0/415.3;break;
            case 'I':cod=F0/440.2;break;
            case 'J':cod=F0/466.8;break;
            case 'K':cod=F0/490.0;break;
            case 'L':cod=F0/515.0;break;
            case 'M':cod=F0/540.8;break;
            case 'N':cod=F0/567.6;break;
            case 'O':cod=F0/596.4;break;
            case 'P':cod=F0/627.2;break;
            case 'Q':cod=F0/659.3;break;
            case 'R':cod=F0/692.8;break;
            case 'S':cod=F0/727.9;break;
            case 'T':cod=F0/764.6;break;
            case 'U':cod=F0/803.0;break;
            case 'V':cod=F0/843.2;break;
            case 'W':cod=F0/885.1;break;
            case 'X':cod=F0/928.8;break;
            case 'Y':cod=F0/974.3;break;
            case 'Z':cod=F0/1021.5;break;
            case ' ':cod=F0/1070.0;break;
            case ',':cod=F0/1120.0;break;
            case '.':cod=F0/1171.9;break;
            case '/':cod=F0/1226.5;break;
            case '\\':cod=F0/1283.8;break;
            case '<':cod=F0/1343.9;break;
            case '>':cod=F0/1406.8;break;
            case '=':cod=F0/1472.5;break;
            case '_':cod=F0/1541.9;break;
            case '~':cod=F0/1614.1;break;
            case '!':cod=F0/1689.1;break;
            case '@':cod=F0/1766.9;break;
            case '#':cod=F0/1847.5;break;
            case '$':cod=F0/1931.0;break;
            case '%':cod=F0/2017.4;break;
            case '^':cod=F0/2106.7;break;
            case '&':cod=F0/2198.9;break;
            case '*':cod=F0/2294.1;break;
            case '&#39;':cod=F0/2392.3;break;
            case '&#34;':cod=F0/2493.5;break;
            case '&#38;':cod=F0/2597.7;break;
            case '&#36;':cod=F0/2704.9;break;
            case '&#33;':cod=F0/2815.1;break;
            case '&#32;':cod=F0/2928.3;break;
            case '&#31;':cod=F0/3044.5;break;
            case '&#30;':cod=F0/3163.7;break;
            case '&#29;':cod=F0/3285.9;break;
            case '&#28;':cod=F0/3411.1;break;
            case '&#27;':cod=F0/3539.3;break;
            case '&#26;':cod=F0/3670.5;break;
            case '&#25;':cod=F0/3804.7;break;
            case '&#24;':cod=F0/3941.9;break;
            case '&#23;':cod=F0/4082.1;break;
            case '&#22;':cod=F0/4225.3;break;
            case '&#21;':cod=F0/4371.5;break;
            case '&#20;':cod=F0/4520.7;break;
            case '&#19;':cod=F0/4672.9;break;
            case '&#18;':cod=F0/4828.1;break;
            case '&#17;':cod=F0/4986.3;break;
            case '&#16;':cod=F0/5147.5;break;
            case '&#15;':cod=F0/5311.7;break;
            case '&#14;':cod=F0/5478.9;break;
            case '&#13;':cod=F0/5649.1;break;
            case '&#12;':cod=F0/5823.3;break;
            case '&#11;':cod=F0/5999.5;break;
            case '&#10;':cod=F0/6178.7;break;
            case '&#9;':cod=F0/6360.9;break;
            case '&#8;':cod=F0/6546.1;break;
            case '&#7;':cod=F0/6734.3;break;
            case '&#6;':cod=F0/6925.5;break;
            case '&#5;':cod=F0/7119.7;break;
            case '&#4;':cod=F0/7316.9;break;
            case '&#3;':cod=F0/7517.1;break;
            case '&#2;':cod=F0/7720.3;break;
            case '&#1;':cod=F0/7926.5;break;
            case '&#0;':cod=F0/8135.7;break;
        }
        mysound();
        printf("%ld\n",cod);
    } while(ch!=27);
}
```

Programming Choices

Scratch – a visual programming environment.



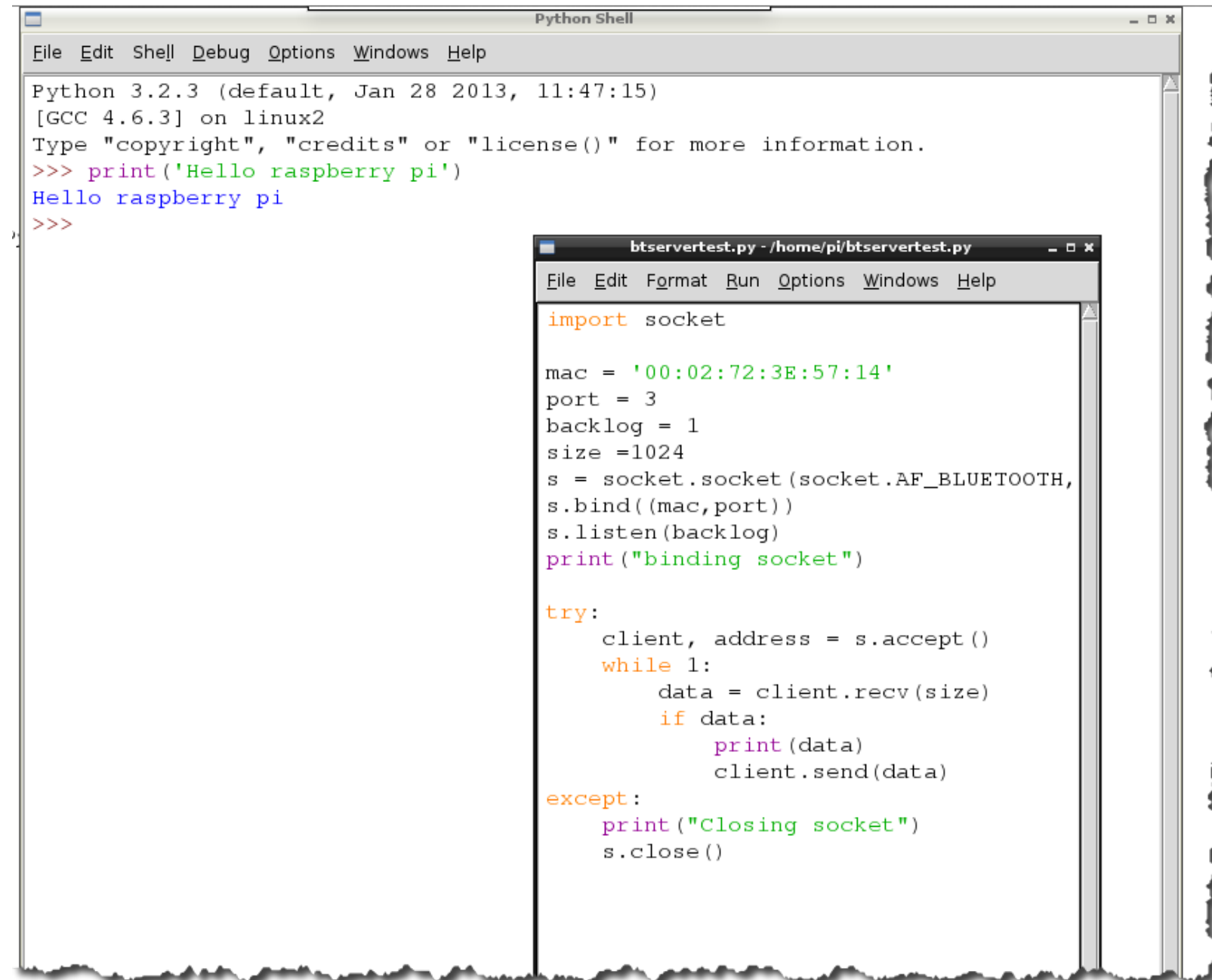
OCR
OCR Resources

See Dr. Joe Hummel's course on programming with Scratch:
<http://www.pluralsight.com/courses/learning-programming-scratc>

Programming Choices

Python – a general-purpose programming language.
Designed to emphasize code readability.

IDLE – a Python environment that ships inside of Raspbian.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.3 (default, Jan 28 2013, 11:47:15)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> print('Hello raspberry pi')
Hello raspberry pi
>>>
```

```
btservertest.py - /home/pi/btservertest.py
File Edit Format Run Options Windows Help
import socket

mac = '00:02:72:3E:57:14'
port = 3
backlog = 1
size = 1024
s = socket.socket(socket.AF_BLUETOOTH,
s.bind((mac, port))
s.listen(backlog)
print("binding socket")

try:
    client, address = s.accept()
    while 1:
        data = client.recv(size)
        if data:
            print(data)
            client.send(data)
except:
    print("Closing socket")
    s.close()
```

Python Basics

- Object-oriented language that supports multiple paradigms of development
 - aspect-oriented, functional (limited)
- Dynamically typed
 - But also strongly typed
- Whitespace is used to denote blocks of code
 - (instead of C-style {brackets})
- Available on almost all platforms
- Used for everything from utilities to web servers
- Created in 1989
 - 3.0 released in 2008
 - 2.7 still very popular

Comments

- Line that begins with the # (hash) character and ends with a line break

```
#this is a comment in Python (at least 3.X)  
#older versions had other comment syntax
```

comments

Types

```
#double quotes  
str = "this is a string"  
#single quotes  
str2 = 'this is also a string'
```

strings

```
#integer  
num = 42  
num = int('42')  
#float  
pi = 3.14  
pi = float('3.14')
```

numbers

```
#null  
n = None  
#True/False  
isNull = True  
isNotNull = False
```

constants

Format Strings

```
#positional replacement tokens  
s = '{0} is {1}'  
#format method to replace tokens  
sf = s.format('foo',42)
```

new-style (Python3)

Comparison

```
#logical and  
foo and bar  
#or  
foo or bar  
#negation  
not foo
```

logical

```
#object identity  
same = foo is bar  
#non identity  
notsame = foo is not bar
```

identity

Lists and Collections

```
#Lists
arr = [42, 43, 44, 45]
print(len(arr)) #prints 4
arr.append(46)
arr.append('Hello')#heterogeneous
arr.extend(1,'2')
;
```

lists

```
#dict
instructors = {'jon':1, 'fritz':2, 'matt':3}
instructors ['aaron'] = 0
print(instructors['jon']) # prints 1
```

dictionary

Indentation for Control Flow

- Code blocks in Python are delimited by whitespace
- Line-feed indicates end of statement
- Indentation (spaces or tab) indicates block

```
if(true){functionCall();  
}  
else  
{  otherFunction();  
}
```

C-style

```
if True:  
    functionCall()  
else:  
    otherFunction()
```

Python

Iterating Collections

- for keyword

```
#Lists  
for n in arr:  
    print(n)  
#prints 42 43 44 45
```

lists

```
#dictionary  
for k,v in instructors.items():  
    print('{0} - {1}'.format(v,k))  
#prints 1 - jon - etc.
```

dictionary

Error Handling

- try/except <Exception>/finally

```
try:
hello(42) #call functions etc
except ArgumentError as err:
print('argument error {0}'.format(err))
except (RuntimeError,TypeError) as err:
print('other error {0}'.format(err))
except:
print('unexpected error {0}'.format( sys.exc_info()[0]))
raise #re-raise the exception
finally:
cleanup()
```

try/except

Functions

- def keyword introduces a named function

```
def hello()  
    print(42)  
def helloWithArg(text)  
    print('hello {}'.format(text))
```

basic

Classes

```
class Util:  
    def hello()  
        print(42)  
    def helloWithArg(text)  
        print('hello {}'.format(text))
```

basic class

Import

- import statement brings in additional libraries
- Many libraries are part of the standard Python distribution
- Likely you will install additional libraries as well

```
import Decimal  
val = Decimal(42.02)
```

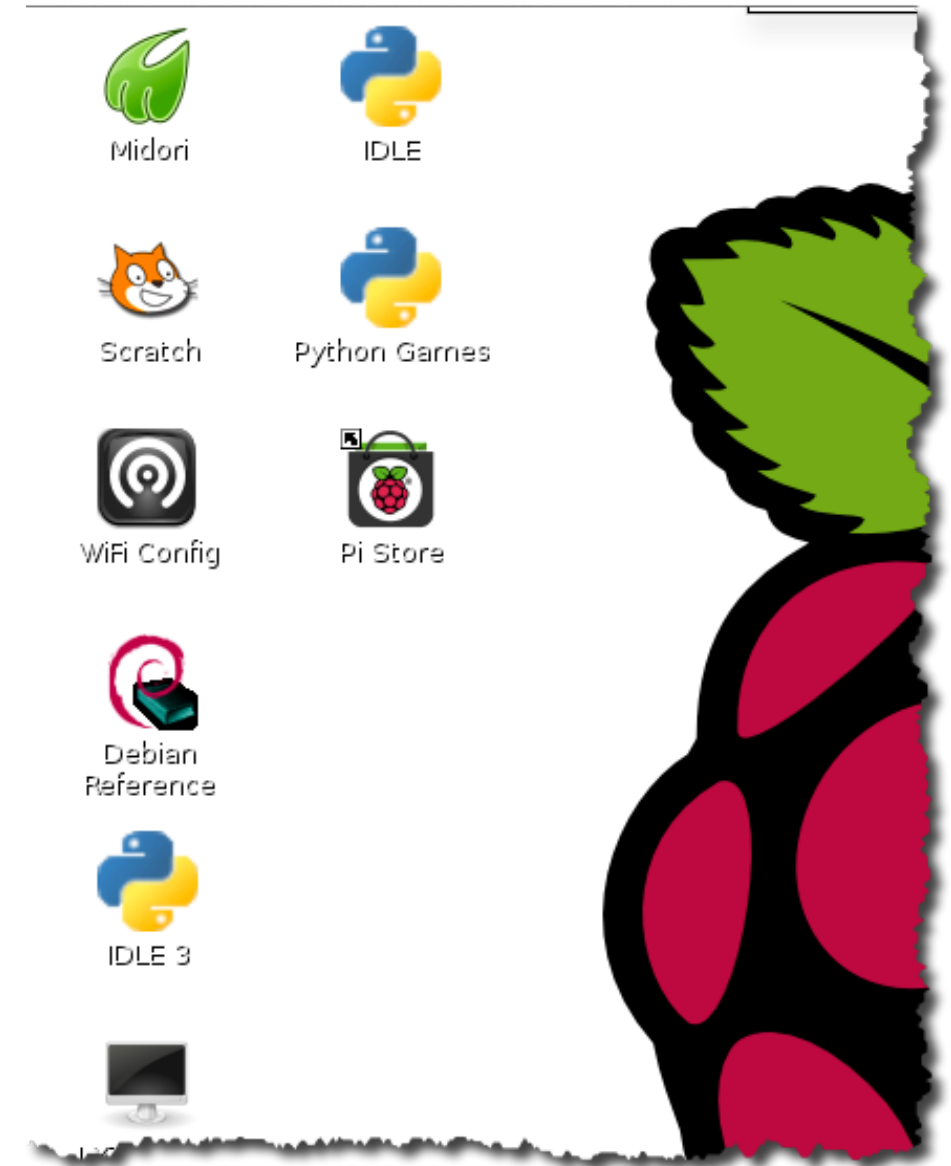
Decimal
Library

```
import json  
val = json.load({42})
```

json Library

IDLE

- Default Python IDE included in Raspbian
- IDLE3 for Python version > 3
- IDLE for Python version < 3
- Both include interactive shell as well as text editor



Demo

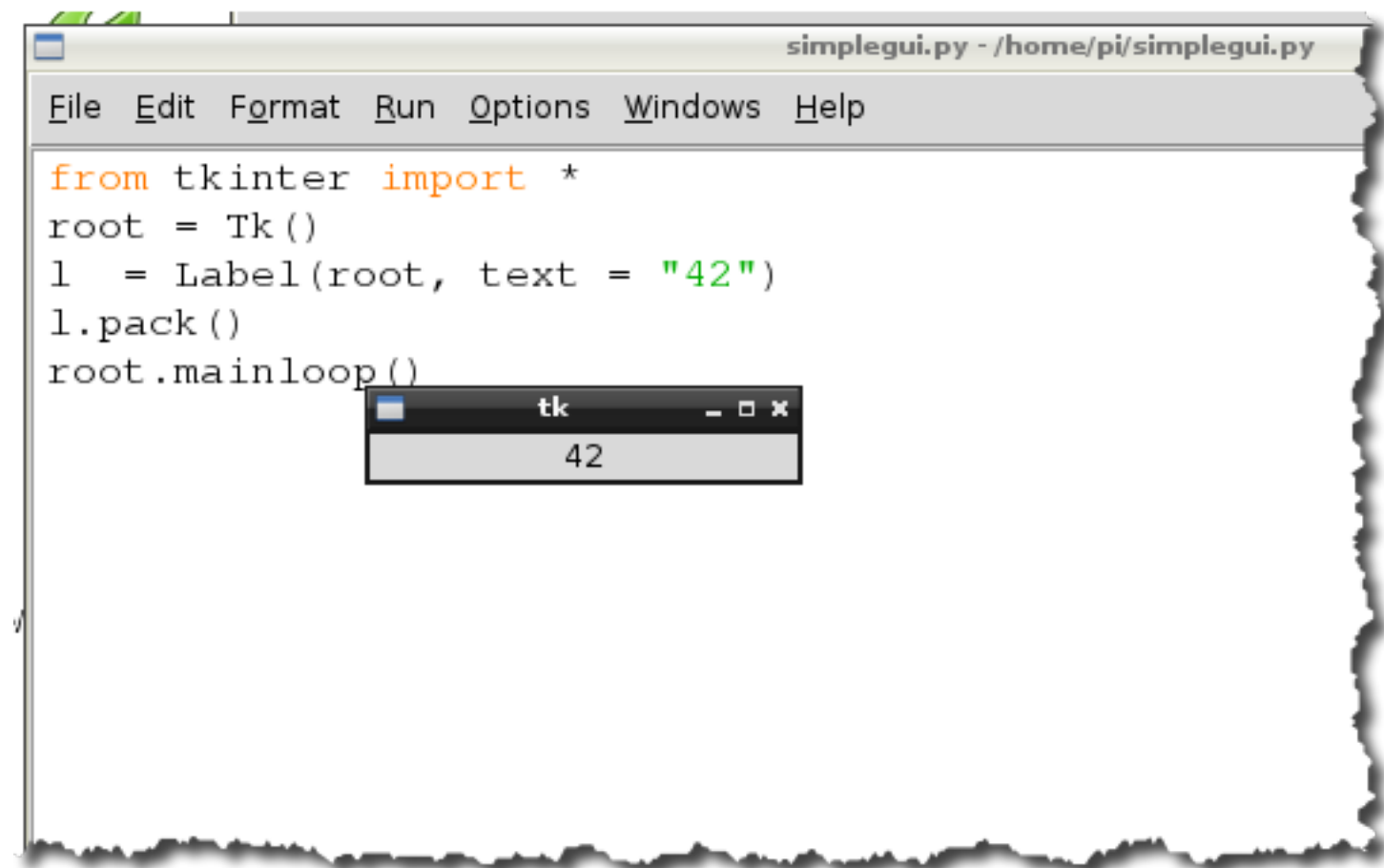
- something cool in python - pull from twitter?

Python and Raspberry Pi

- You may need extra Python libraries to work with specific hardware with the Pi
 - E.g. bluetooth (covered in the wearable module)
- RPi.GPIO now included by default in Raspbian
 - Library to control GPIO pins
 - We'll cover this in the next module

Python and X11

- You might want to create a GUI program with Python
- There are many Python GUI libraries
 - For the Raspberry Pi Tkinter is the de facto standard



Demo

- twitter gui?

C/C++

- gcc is installed by default on Raspbian
- You can write native code to your heart's content
- More complex usage model for typical Raspberry Pi use cases
- Remember - you can write Python modules in C and create wrappers for Python programs
 - Probably the best option if you have to wrap something low-level
- You might end up using gcc when installing certain packages that need to be built on the platform
 - We will do this in a later module for Node.js

bash

- Of course you can always use bash to do simple to complex “programs”
- You can even integrate with hardware
 - See the next module

Summary

- As an open platform Raspberry Pi gives you many choices for writing code
- Python is the most supported (and most documented)
- C/C++ if you need to get close to the metal