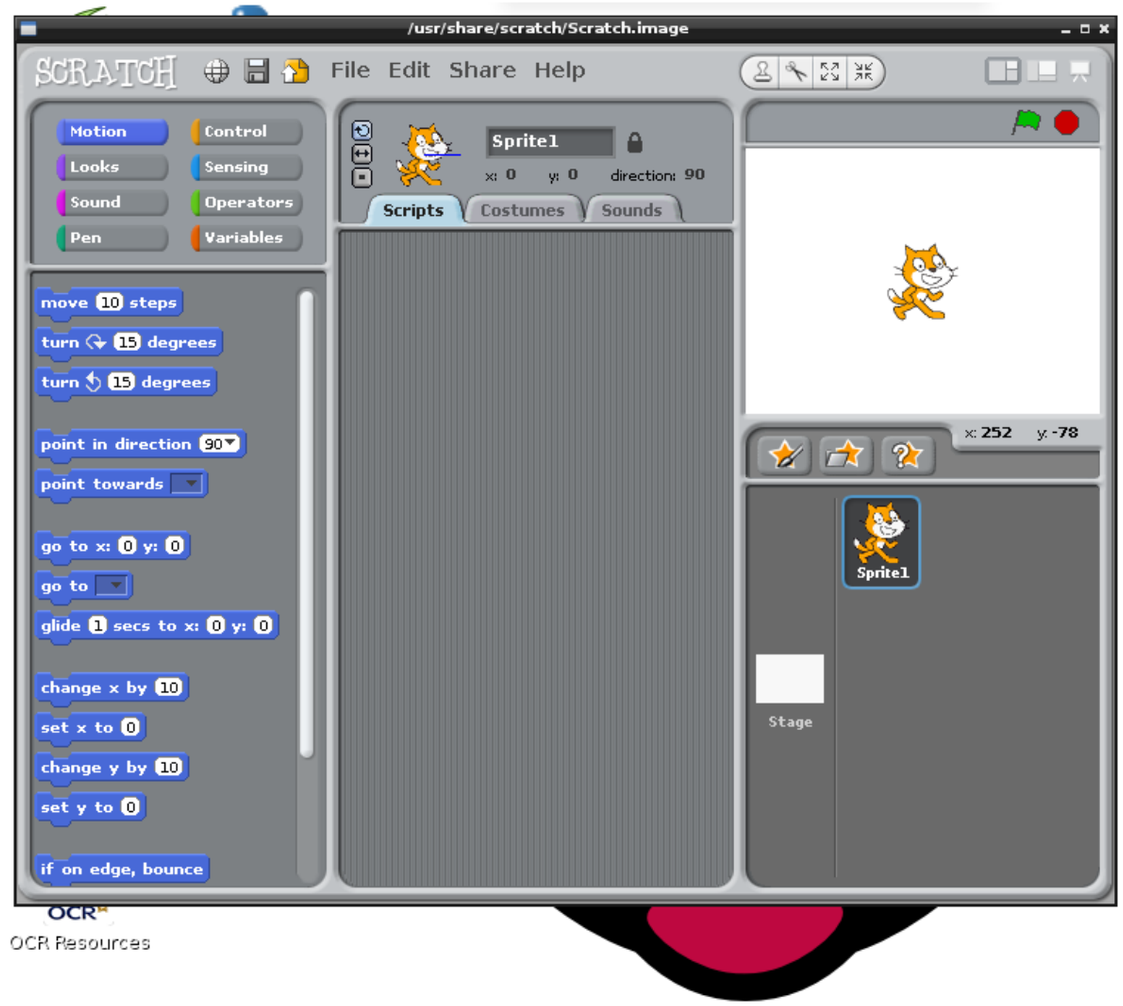
Programming the Raspberry Pi

Jon Flanders
@jonflanders

This module is about..

Programming Choices

Scratch – a visual programming environment.



OCR Resources

See Dr. Joe Hummel's course on programming with Scratch:
ttp://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
nisNull = True
nisNotNull = 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 {0}’.format(text))
```

basic

Classes

```
class Util:
def hello()
    print(42)
def helloWithArg(text)
    print(‘hello {0}’.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)
```

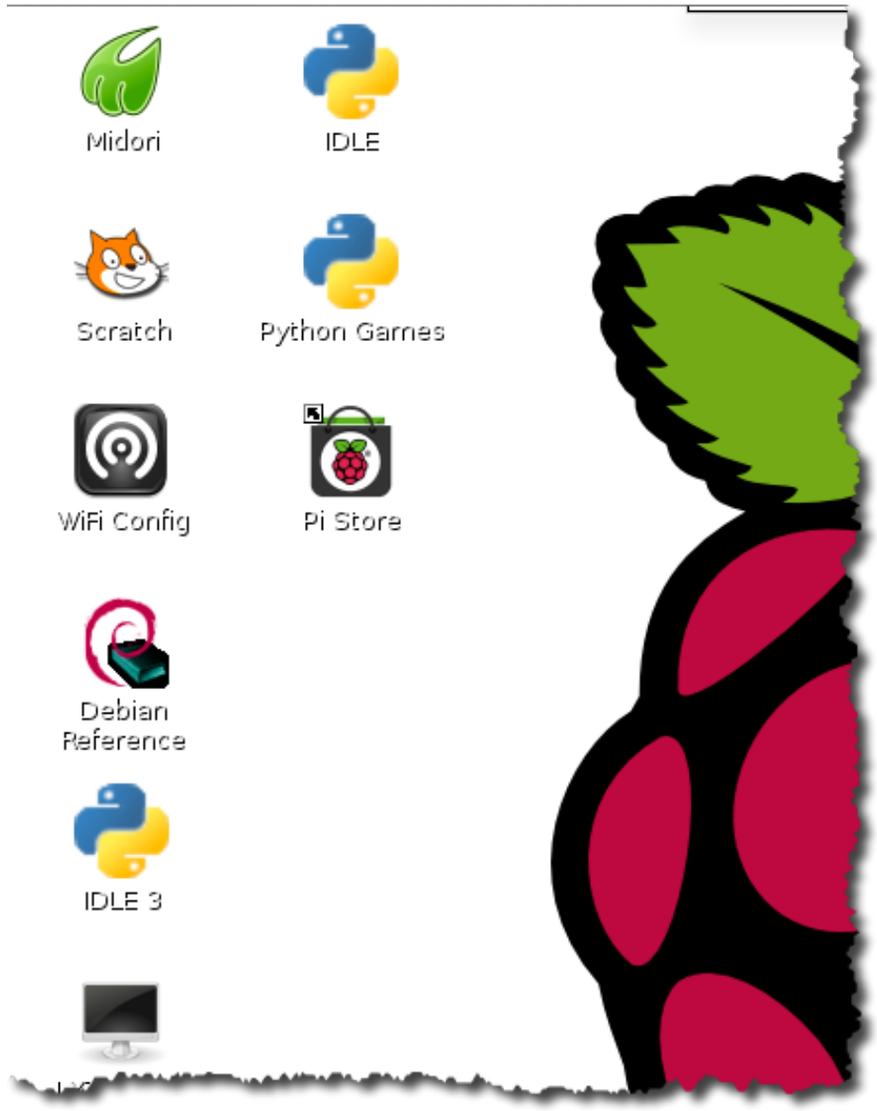
Decimal Library

```
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

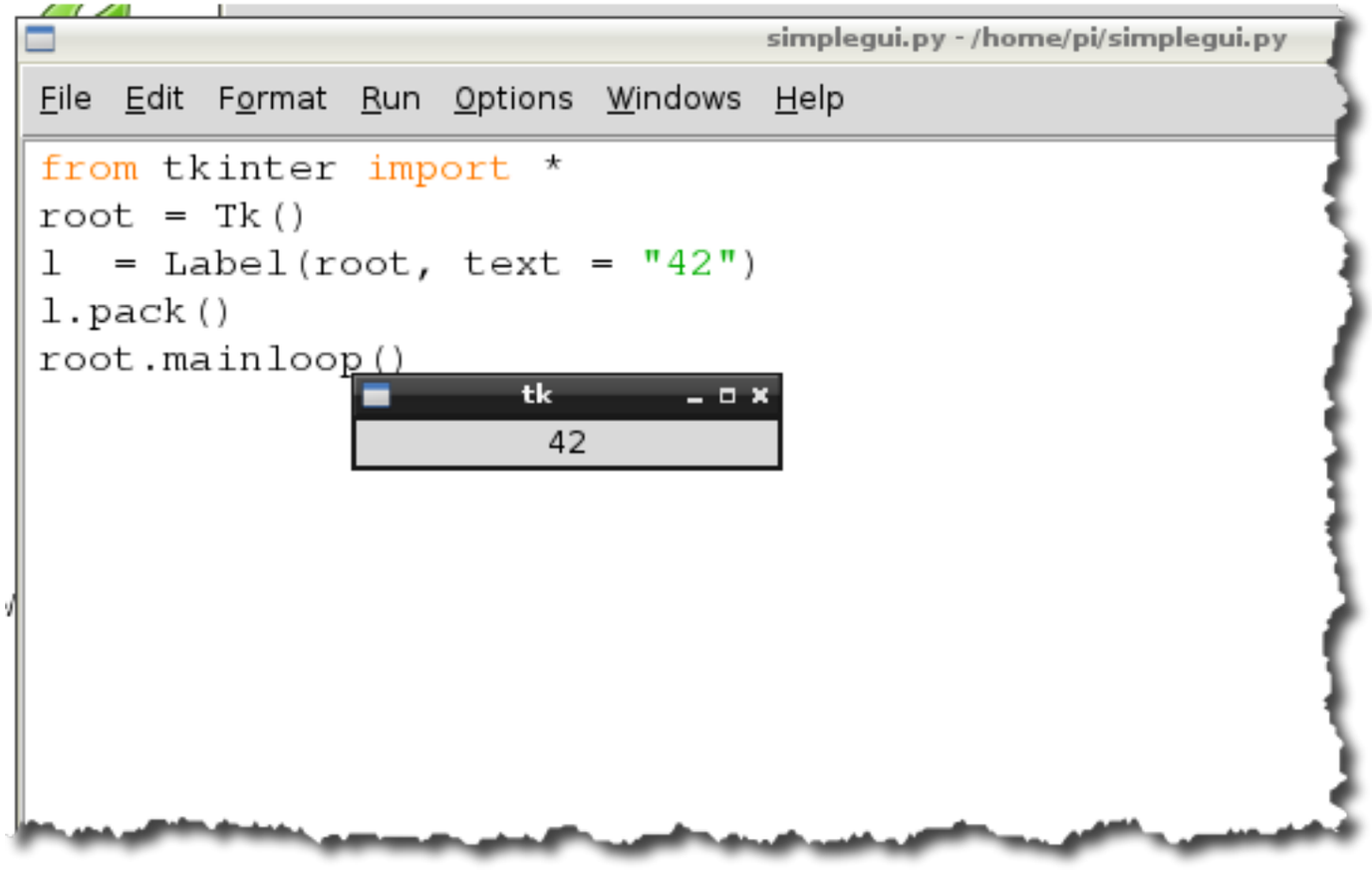
- something cool in python - pull from twitter?

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