IoT

Internet of things

Connecting devices and making them smart

30 million devices connected by 2020. ABI Research

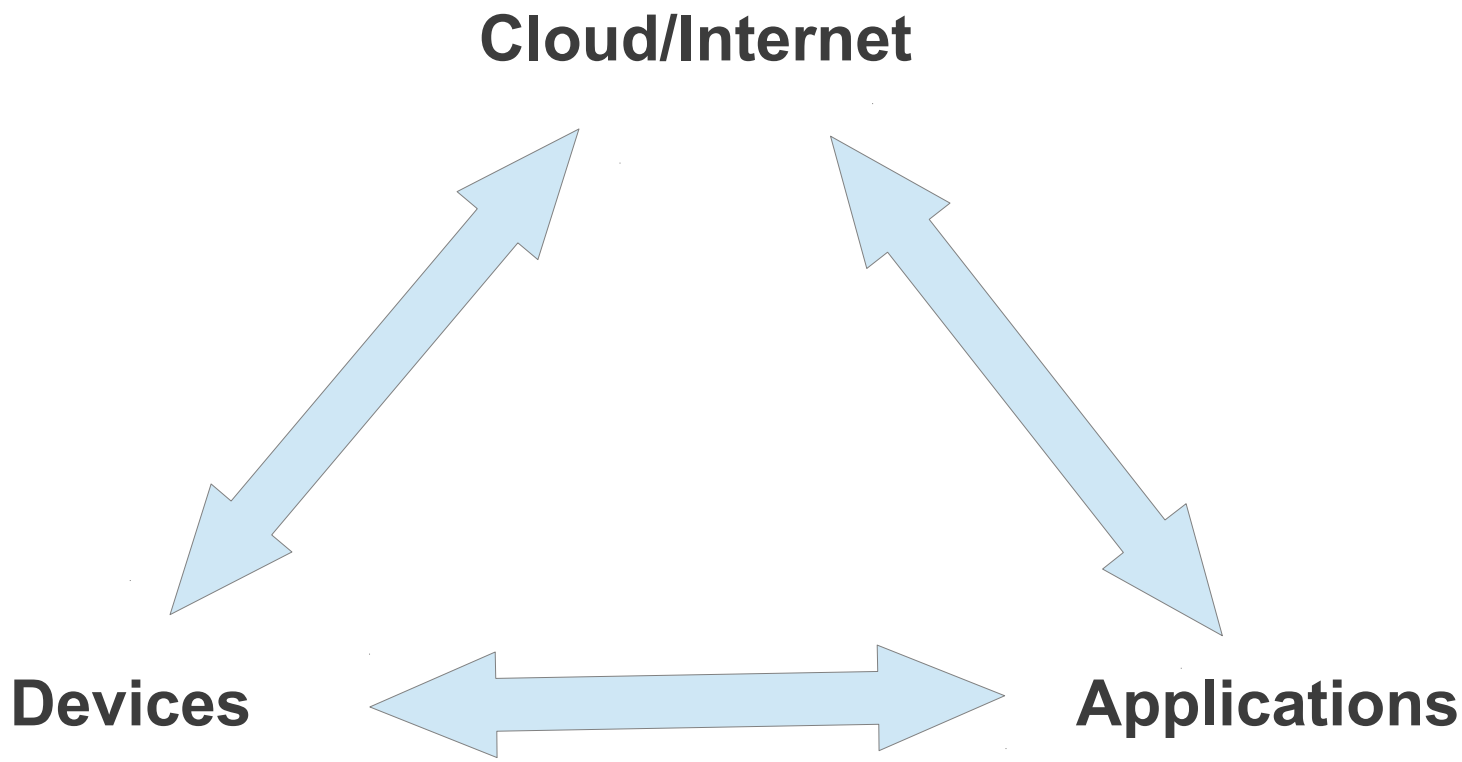
Cisco estimates that 50 billion devices and objects will be connected to the Internet by 2020

Yet today, more than 99 percent of things in the physical world remain unconnected.

What is Internet of Things??

The IoT is the next technology transition when devices will allow us to sense and control the physical world. It's also part of something even bigger.

The ability to network embedded devices with limited CPU, memory and power resources means that IoT finds applications in nearly every field.



Devices

A power control device with two plug ins.

Temerature sensing device.

Light Intensity sensing device.

Cloud/Internet

Using a cloud PARSE.

Making a database.

Creating a client code to connect to device gateway and your application.

Application

Making an app on Kivy.

Testing it on your system (PC).

Making an apk for Android.

Can also be converted for an iOS app.

What all to learn?????

Devices

Arduino progamming
I/O operations
Serial communication

Connecting relay for AC control
Code testing

Setting Bluetooth module
Testing communication with PC

Arduino Programming

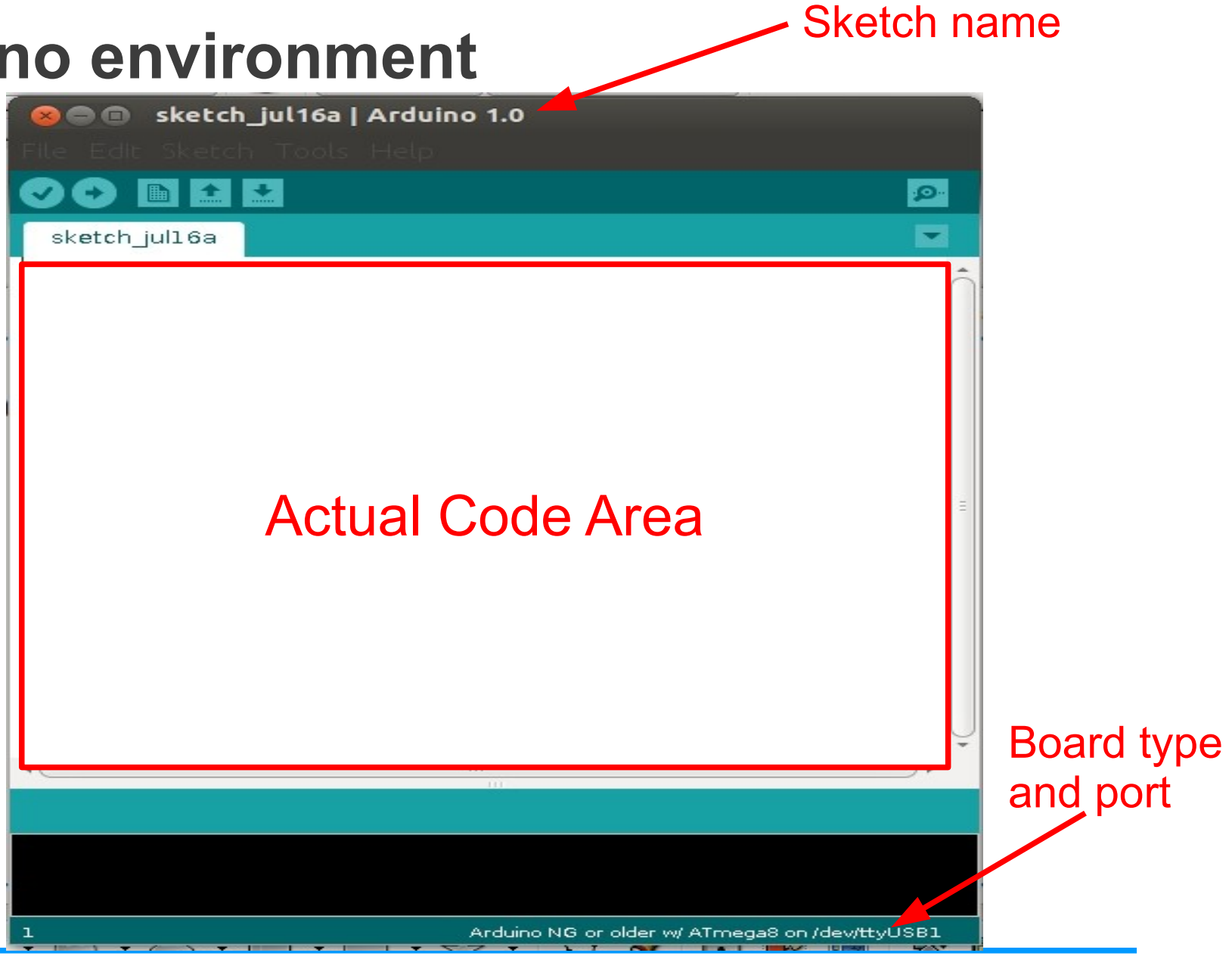
Download and install Arduino IDE

www.arduino.cc

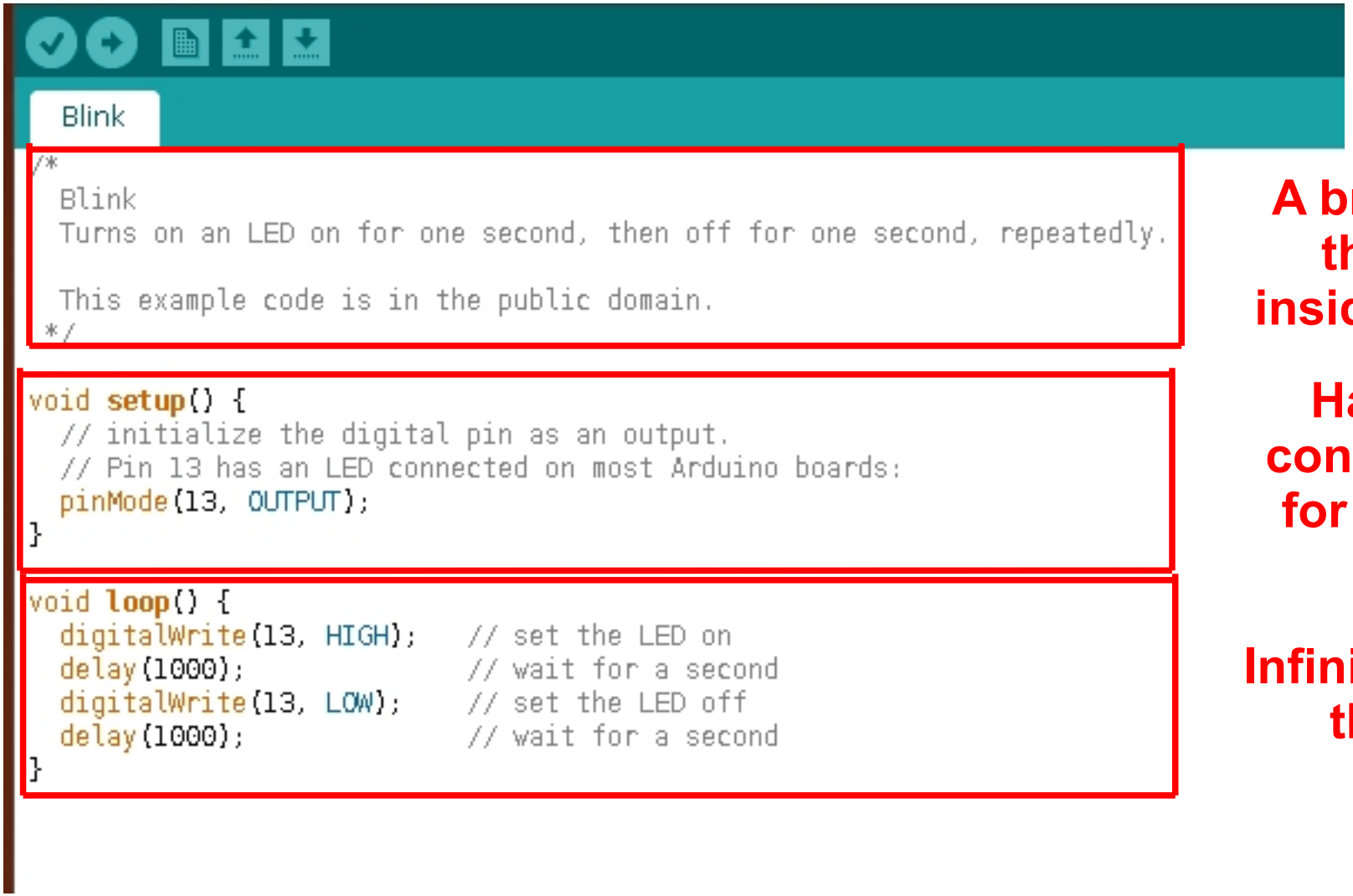
Select your OS and download the package and follow the instruction given.

The package is also provided with the material given to you. Follow the Arduino installation manual given inside it.

Arduino environment



Blink (File<<Examples<<Basics<<Blink)



A brief about the code inside /*-------*/

Hardware configuration for one time run

Infinite loop for the task

PinMode()

Description

Configures the specified pin to behave either as an input or an output.

Syntax

pinMode(pin, mode)

Parameters

pin: the number of the pin whose mode you wish to set

mode: either INPUT or OUTPUT

Returns

None

digitalWrite()

Description

Write a HIGH or a LOW value to a digital pin.

Syntax

digitalWrite(pin, value)

Parameters

pin: the pin number

value: HIGH or LOW

Returns

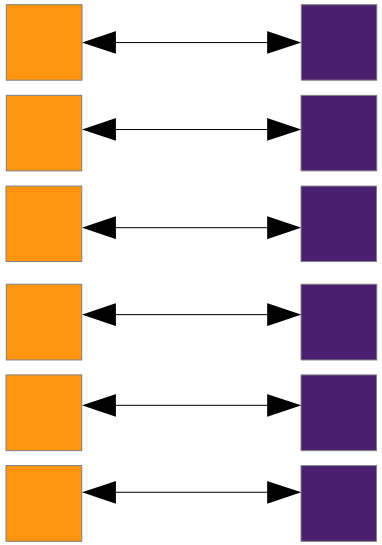
none

How to download the code in Fruduino

Open
IoT_workshop<<doc<<Fruduino_install.pdf

Online refernce

Serial & Parallel Communication



Parallel Communication

Serial Communication

Speed at which the communication happens. **Baudrate**.

IoT_workshop<<Code<<Arduino<<SerialRead<<SerialRead.ino

SerialRead

```
int incomingByte = 0;    // for incoming serial data
int led = 13;
void setup() {
        Serial.begin(9600);
        pinMode(led,OUTPUT);
}

void loop() {
  // send data only when you receive data:
  if (Serial.available() > 0) {
    // read the incoming byte:
    incomingByte = Serial.read();
    if (incomingByte == 'y'){
      digitalWrite(led,HIGH);
      println("LED high");
    }
    if (incomingByte == 'n'){
      digitalWrite(led,LOW);
      println("LED low");
    }
  }
}
```

Serial port set to (9600 baudrate)

If new data is there to read

Read data in serial buffer

Printing the sensor value to the serial port

begin()

Description

Sets the data rate in bits per second (baud) for serial data transmission.

Syntax

Serial.begin(speed)

Parameters

speed: in bits per second (baud) - long

Returns

nothing

available()

Description

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 128 bytes). available() inherits from the Stream utility class.

Syntax

Serial.available()

Parameters

none

Returns

the number of bytes available to read

println()

Description

Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). This command takes the same forms as Serial.print().

Syntax

Serial.println(val)

Serial.println(val, format)

Parameters

val: the value to print - any data type

format: specifies the number base (for integral data types) or number of decimal places (for floating point types)

Returns

byte

Analog to Digital

Analog range: 0 V to 5 V

Digital: 10 bit

$2^{10} = 1024$

Digital range: 0 to 1023

AnalogReadSerial

(File<<Examples<<Basics<<AnalogReadSerial)

AnalogReadSerial

```
/*
  AnalogReadSerial
 Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

Reading the analog value of the sensor at A0 pin

analogRead()

Description

Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter.

Syntax

analogRead(pin)

Parameters

pin: the number of the analog input pin to read from (0 to 5)

Returns

int (0 to 1023)

Temperature sensor (LM35)

As the change in voltage of the sensor is in range 0 to 1V

Arduino has analogReference(INTERNAL); sets to aref to 2.56 V

Analog range: 0 to 2.56 V
Digital range: 0 to 1023

1 division of ADC = 2.56/1023 = 0.0025 V = 2.5 mV
For LM35, 10 mV change is 1°C change, 10/2.5 = 4

So,
For every change of 4 in analog reading, there is 1°C.

IoT_workshop<<Code<<Arduino<<TempSerial<<TempSerial.ino

TempSerial

```
/*
  TempSerial
 Reads analog data from the temperature sensor LM35 on pin 0, prints the result to the serial monitor

 1 division of ADC = 2.56/1000 = 0.0025 V = 2.5 mV
 For LM35, 10 mV change is 1°C change, 10/2.5 = 4

 This example code is in the public domain.
 */
float tempC;
int reading;
int tempPin = 0;

void setup() {
  Serial.begin(9600);
  analogReference(INTERNAL);
}

void loop() {
  reading = analogRead(tempPin);
  tempC = reading / 4;
  Serial.println(tempC);
}
```

declaring variables

aref to 2.56 V from 5 V

Now lets write the code for the device with temperature sensor, and two output SSR and connecting the bluetooth module from the PC.

Cloud/Internet

Making account in www.parse.com

Name your project
Application ID
REST_API key
Master key

Making a client through Python

Connecting to cloud database
Serial communication with Arduino
Setting up of bluetooth module

Python

- Powerful
- Dynamic
- Object oriented programming language

Why Python??

- **Portability**
  - Linux/Unix
  - Windows
  - Mac
  - OS 2 and others
- **Integration**
  - COM
  - .NET IronPython
  - COBRA
  - JAVA Jython
  - C&C++ Cython

- **Easy**
  - Begin
  - Clean
  - Readable syntax
  - Making applications becomes very simple
- **Power**
  - New extensions written all the time
  - App dev, web development, GUI, Game dev etc
  - Audio/ Video editing, database access
- **Dynamic**
  - Most flexible language
  - Creative
  - Solves design and development issues

- **Open Source**
  - Creation is free
  - Distribution is also free

Invoking the Interperter

```
Python 2.7.3 (default, Apr 10 2013, 05:46:21)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Prompt is indicated by “>>>”

If the execute command requires more input, a `...` prompt will be displayed.

- **Execute Python Statement**

```
>>> print “Hello World”
Hello World
```

- **Running scripts from command line**

```
~$ python script.py
Executing a Script
~$
```

- **Running scripts from interpreter**

```
>>> execfile(script.py)
Executing a Script
>>>
```

Code Indentation

```
if True:
    print "true"
else:
    print "false"
```

error

```
if True:
    print "true"
    print "Answer"
else:
    print "false"
       print "Answer"
```

Multiple Statements

```
total_sum = item_1+ \
item_2\
item_3
```

Quotation

```
word= 'String'
Sentence= “This is a String”
Paragraph= “””This is a String. It will have multiple lines.”””
```

Formating String

```
For string %s
For number %d
```

```
>>> print "%s/%s/%d" %("Anirban","Jodhpur",2014)
```

Control Statement

If elif else

```
if x>10:
    print "x is large"
elif y>10:
    print "y is large"
else:
    print "x and y are small"
```

while

```
x = 1
while x<10:
   print “Happy days”
```

For

```
word = “Python”
for ch in word:
   print ch
```

Data types

List

```
newList=[200,"python",5]
```

Truple

```
newTruple=(200,"python",5)
```

Dictionary

```
newDic={1:'one',2:'two',3:'three'}
```

Python
Objects, Modules, Classes and Function

Using Objects

Every piece of data stored and used in the python language is an object.
All objects should in python has:

An identity
A type
A value

```
>>>l= [1,2,3]
>>>print id(l)
9267480
>>>print type(l)
<type 'list'>
>>>print l
[1,2,3]
```

identity

type

value

Using Modules

Entire Python language is built up of modules. Inbuild modules are Python(or C) files bundled together with the Python language.

You can make your own modules as well as use third party modules.

To load a python module

```
>>> import os
>>> os.path.abspath(“.”)
'C:\\books\\python'
>>> import os as computer
>>> computer.path.abspath(“.”)
'C:\\books\\python'
>>> from os import path
>>> computer.path.abspath(“.”)
'C:\\books\\python'
```

Can use a different name for the module

Can import a module from a package

Function

```
def fun(name, location, year=2006):
   print"%s/%s/%d" % (name, location,year)
```

Default parameter

```
>>>fun("Anir","India")
Anir/India/2006
>>>fun(location="India",year= 2014, name="Anir")
Anir/India/2014
>>>fun("Anir",location="India",year= 2014)
Anir/India/2014
```

Passing parameters in order with default year

Passing parameters by name

Mix different methods for passing parameter

```
>>>tuple = (”Anir”,”Bangalore”,2010)
fun(*tuple)
Anir/Bangalore/2010
>>>dic = ('name':'Anir','location':'Ooty','year':2010)
fun(**dic)
Anir/Ooty/2010
```

Creating a tuple

Passing tuple as a function parameter

Passing dictionary as a function parameter

Creating a dictionary

```
>>>def square(x):
...    return x*x

>>>print square(3)
9
```

Excepts a number as a parameter

Returns square of the number

Calling function with function parameter

Understanding Python Classes

Classes are basically a collection of attributes and methods.

Used for two reason:

To create a whole new user-defined data type.

To extend the capabilities of an existing one.

`__init__()` function overwrites the method inherited.

```
class testClass(object):
   print “Creating New Class\n======================”
   number=5
   def __init__(self, string):
      self.string = string
   def printClass(self):
      print “Number = %d”% self.number
      print “String = %s”% self.string
```

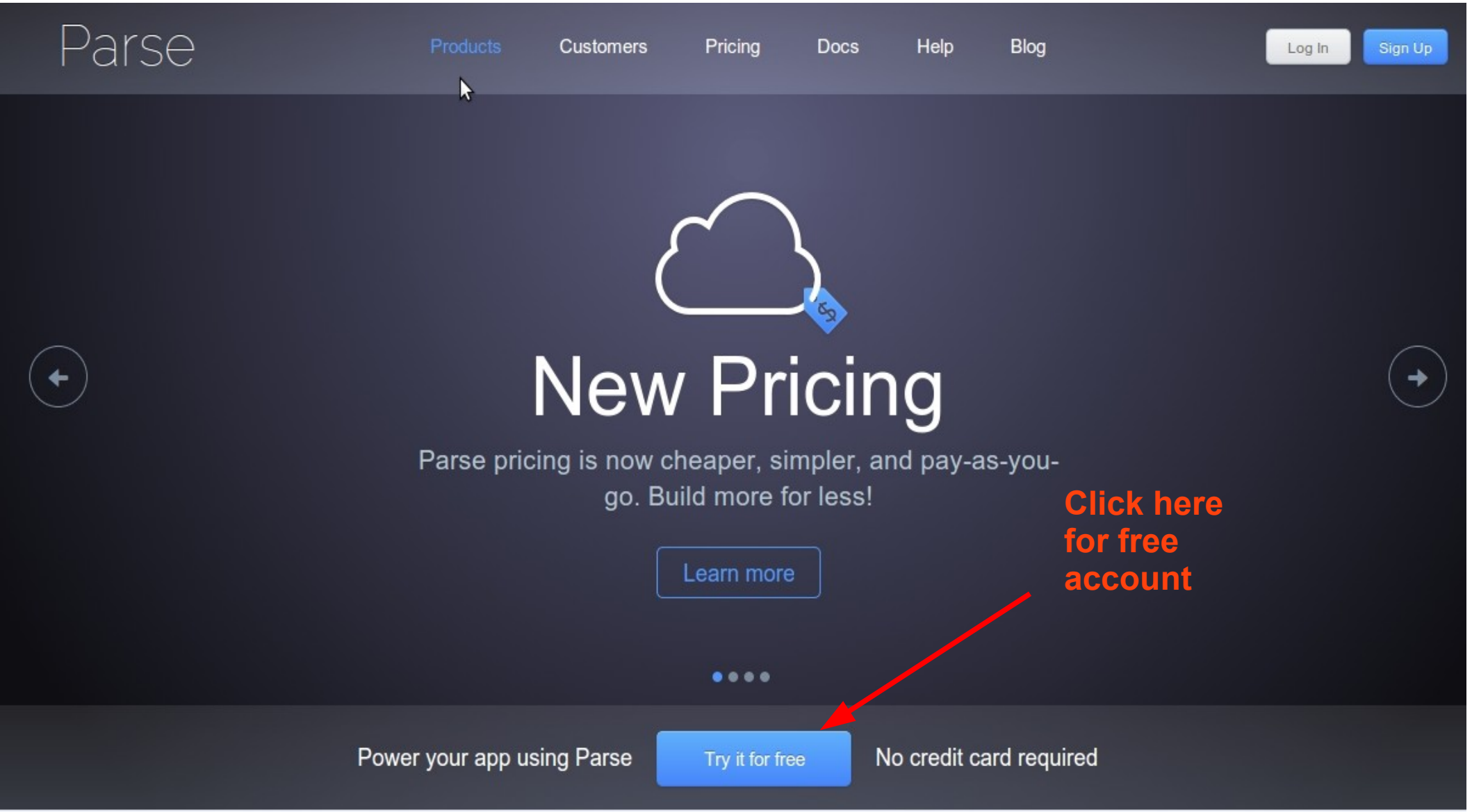
```
tc = testClass(“Five”)
tc.printClass()
tc.number =10
tc.string = “Ten”
tc.printClass()
```

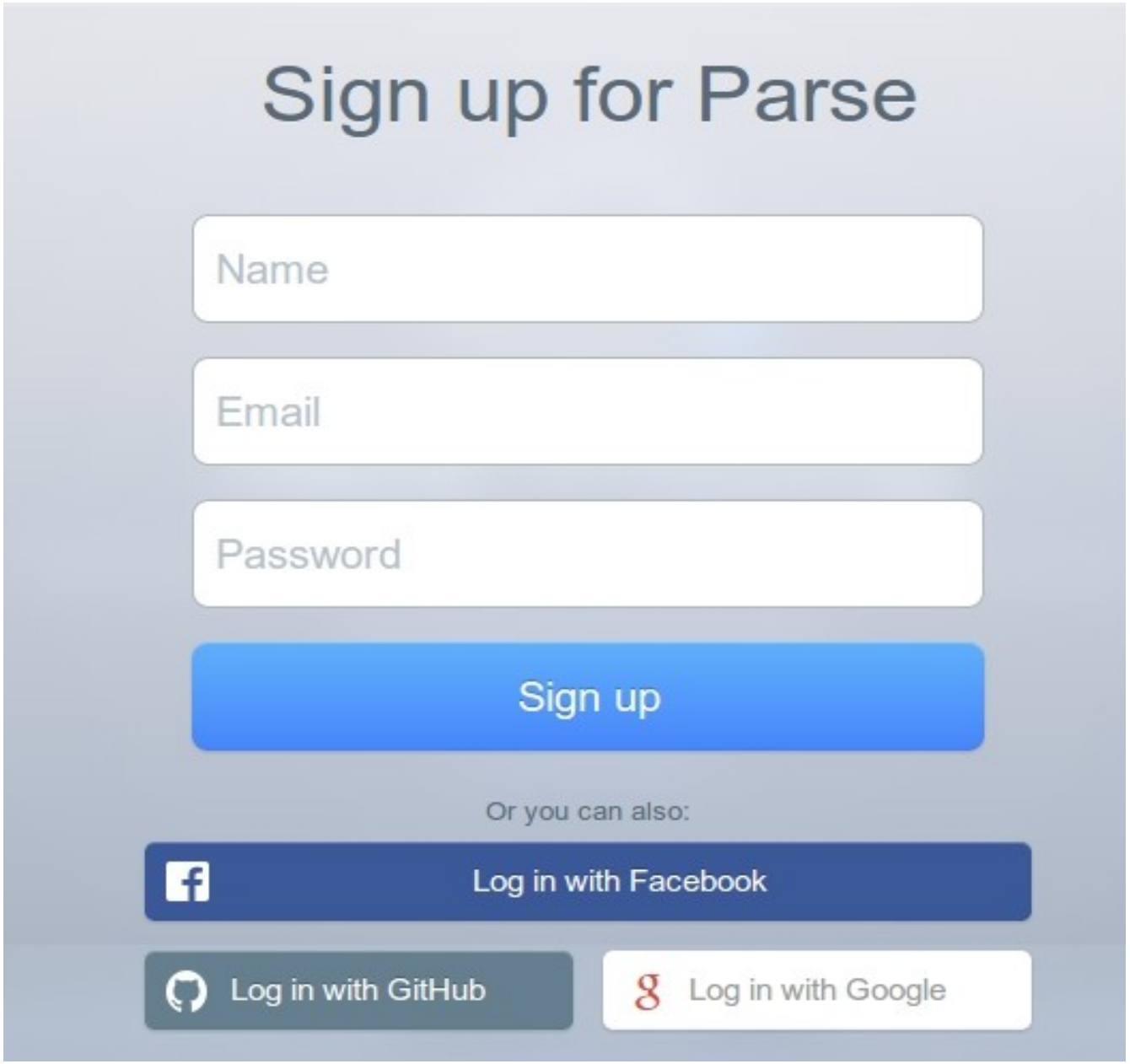
Output

```
Creating New Class
==========================
Number = 5
String = Five
Number = 10
String = Ten
```

Lets start with Cloud

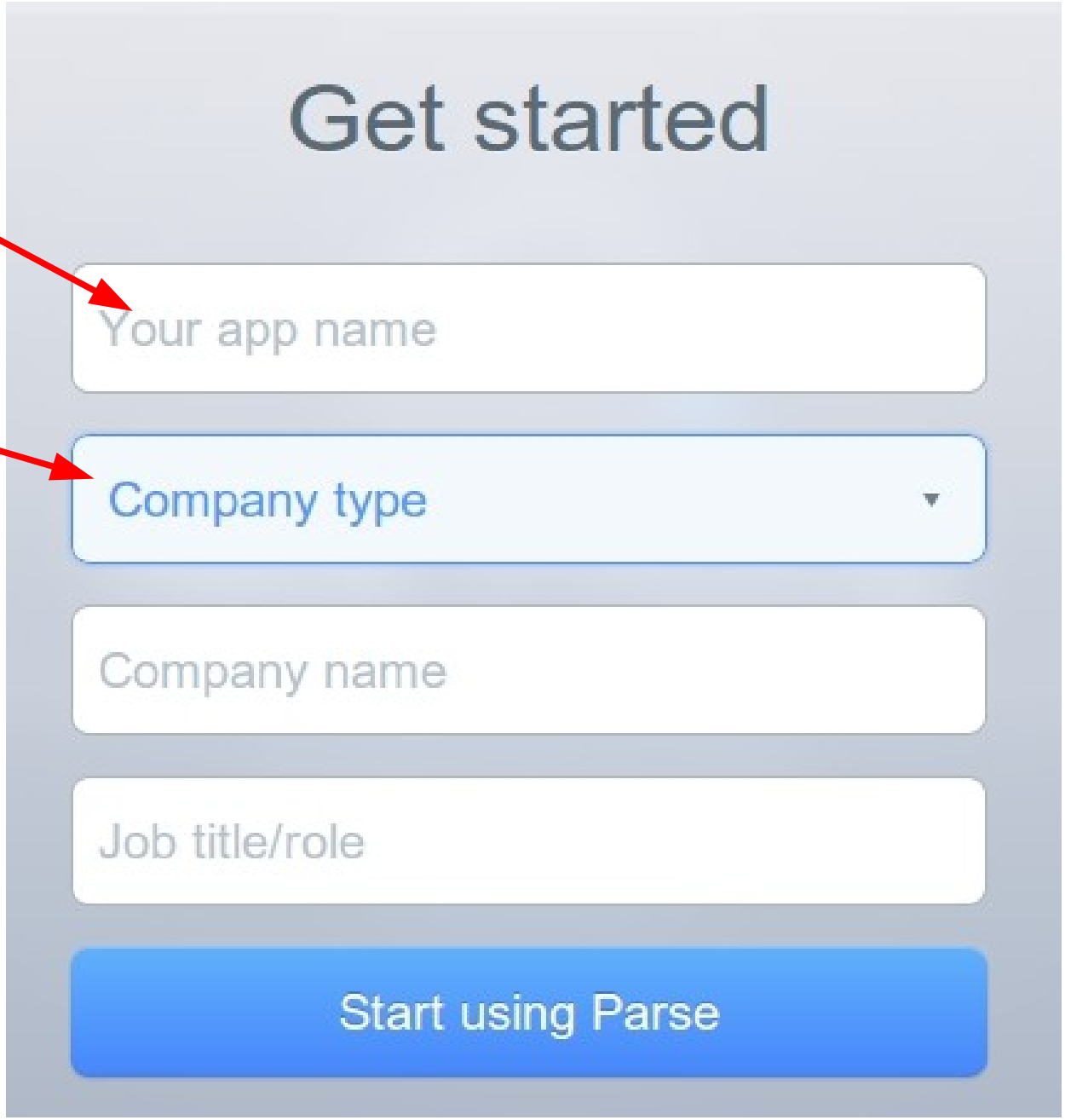
www.parse.com



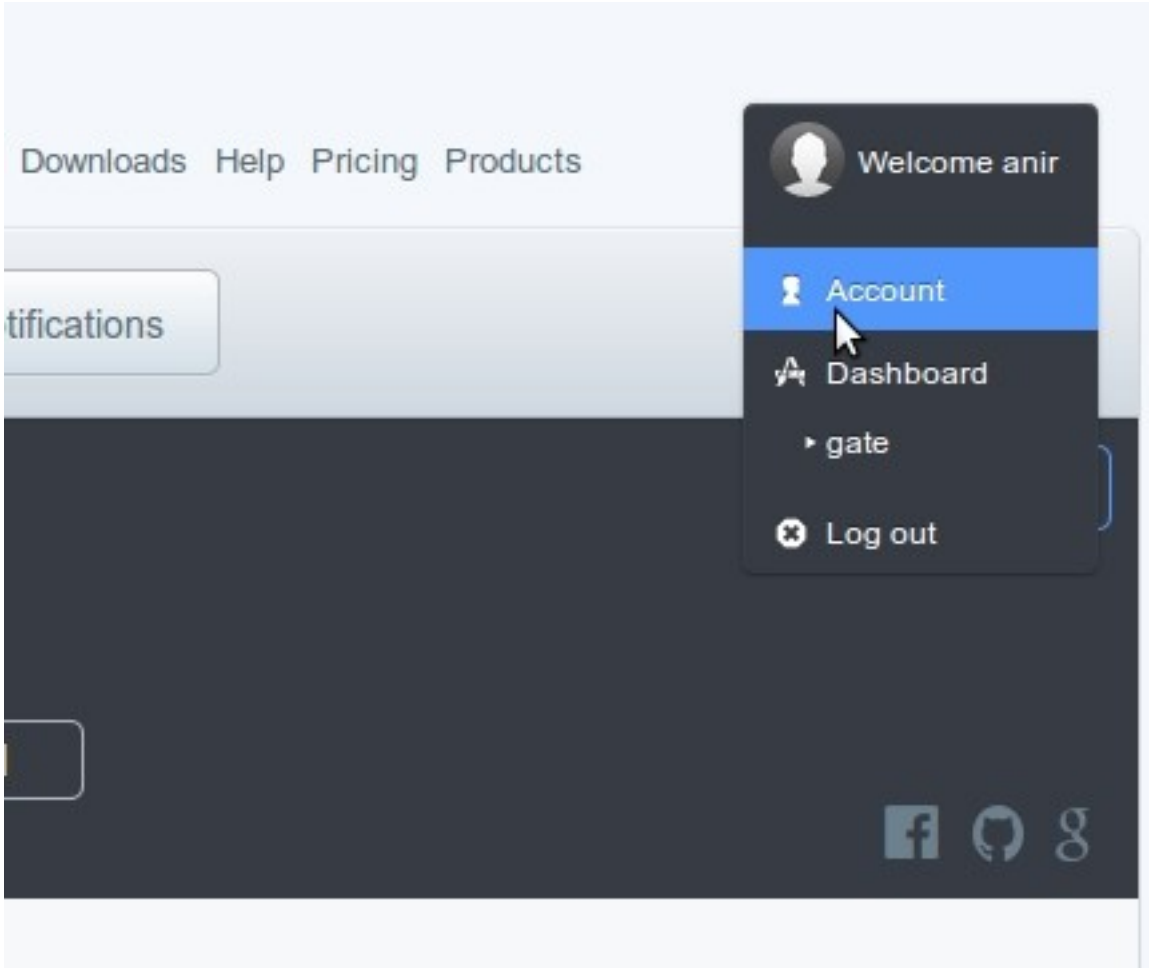


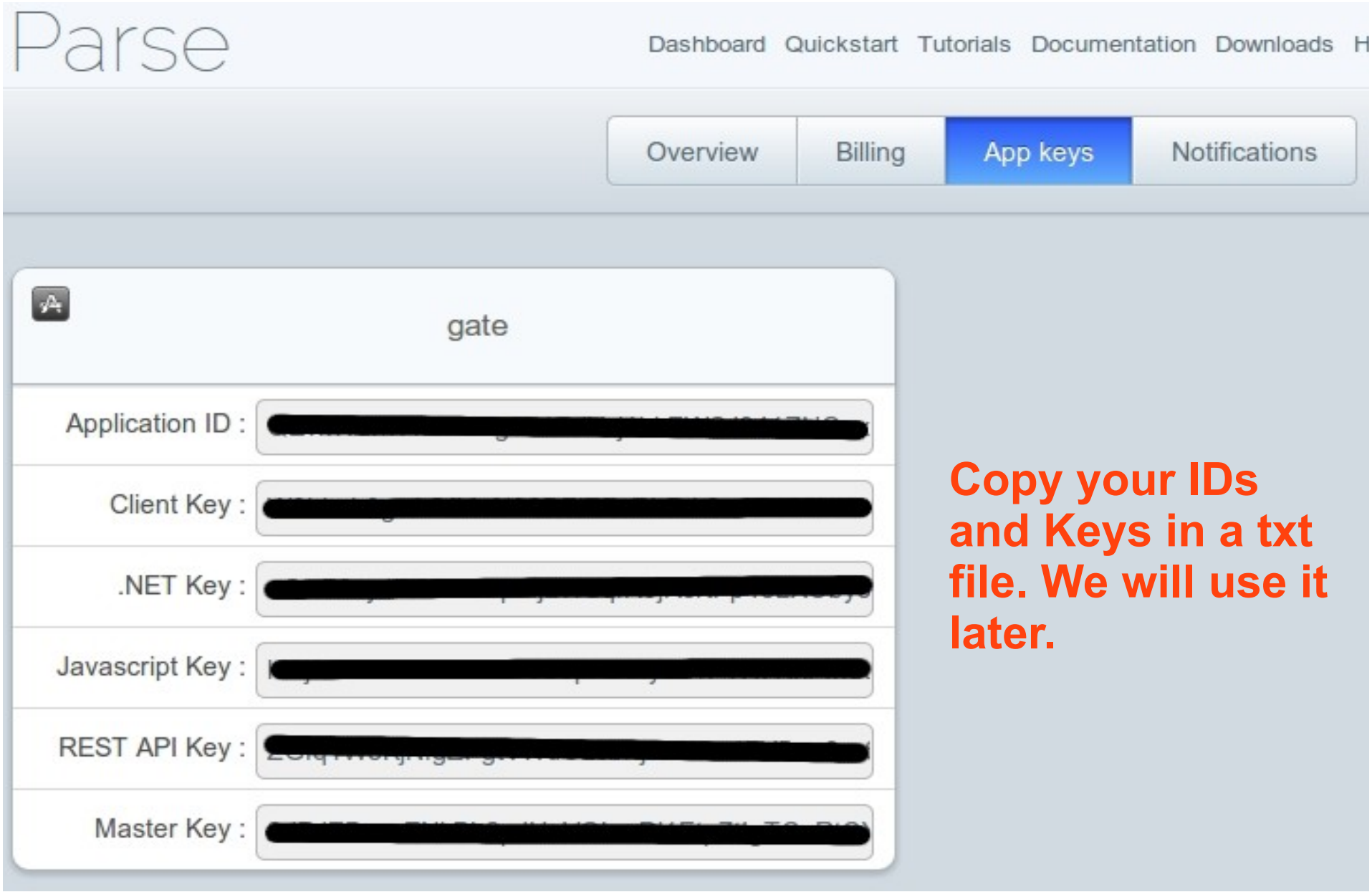
Give a name to your app

Select Individual Developer



On the right get welcome and select Account





Time to start using your just made cloud.

ParsePy

Lets start with the test our cloud connection.

Create a **settings_local.py** file in your local directory with three variables that define a sample Parse application to use for testing:

```
APPLICATION_ID = "APPLICATION_ID_HERE"
REST_API_KEY = "REST_API_KEY_HERE"
MASTER_KEY = "MASTER_KEY_HERE"
```

Create a **test.py**

```
from parse_rest import tests
tests.run_tests()
```

For registration add this lines to your code

Format

```
from parse_rest.connection import register
register(<application_id>, <rest_api_key>[, master_key=None])
```

Example

```
from parse_rest.connection import register
register("**************************","**************************")
```

Get access/make to the database

```
from parse_rest.datatypes import Object

first_object = Object()
```

Lets make a class

```
from parse_rest.datatypes import Object
class Gateway(Object):
    pass
```

And then instantiate it with your parameters:

```
gateway = Gateway(switch_one=0, switch_two=0,temperature=25)
```

To save our new object, just call the save() method:

```
gateway.save()
```

You need to know the objectId so print it and copy it

```
print gateway.objectId
```

Start Querying

```
gate = Gateway.Query.get(objectId="xxwXx9eOec")
```

To change the value

```
gate.switch=1
```

To get the value

```
status = gate.switch
```

To get the all the object

```
all_scores = Gateway.Query.all()
for my in all_status:
    print my.switch_one
    print my.switch_two
    print my.temperature
```

cloud.py

```
try:
    import settings_local
except ImportError:
    sys.exit('You must create a settings_local.py file with'\
APPLICATION_ID, ' \
                'REST_API_KEY, MASTER_KEY variables set')

from parse_rest.connection import register
register(settings_local.APPLICATION_ID,settings_local.REST_API_KEY)

from parse_rest.datatypes import Object

class Gateway(Object):
   pass
gateway = Gateway(switch_one=0, switch_two=0,temperature=25)
gateway.save()
print gateway.objectId
```

full_gate.py

```
try:
    import settings_local
except ImportError:
    sys.exit('You must create a settings_local.py file with '\
'APPLICATION_ID, ' \
                'REST_API_KEY, MASTER_KEY variables set')

from parse_rest.connection import register
register(settings_local.APPLICATION_ID,settings_local.REST_API_KEY)

from parse_rest.datatypes import Object

class Gateway(Object):
   pass

all_status = Gateway.Query.all()
for my in all_status:
   print my.switch_one
   print my.switch_two
   print my.temperature
```

client.py

```
try:
    import settings_local
except ImportError:
    sys.exit('You must create a settings_local.py file with'\
'APPLICATION_ID, ' \
                'REST_API_KEY, MASTER_KEY variables set')

from parse_rest.connection import register
register(settings_local.APPLICATION_ID,settings_local.REST_API_KEY)

from parse_rest.datatypes import Object


class Gateway(Object):

   def GateUpdate_switch_one(self,Id=None,sw=0):
      gate = self.Query.get(objectId=Id)
      gate.switch_one = sw
      gate.save()
```

client.py (contd.)

```
    def GateUpdate_switch_two(self,Id=None,sw=0):
        gate = self.Query.get(objectId=Id)
        gate.switch_two = sw
        gate.save()

    def Status_switch_one(self,Id=None):
        gate = self.Query.get(objectId=Id)
        status = gate.switch_one
        return status

    def Status_switch_two(self,Id=None):
        gate = self.Query.get(objectId=Id)
        status = gate.switch_two
        return status

if __name__=="__main__":
    gate_test= Gateway()
    gate_test.GateUpdate(Id="******")
```

Lets get started with Kivy

Install Kivy on your system with Kivy_install.pdf

Starting with a simple app

As you have not written any thing in the class TutorialApp(App) so you can see a blank screen

Adding a button

Lets add a button function with text 'Hello!' and some background colours

```
from kivy.app import App
from kivy.uix.button import Button

class TutorialApp(App):
    def build(self):
        return
Button(text='Hello!',background_color=(0,0,1,1),fon
t_size=150)

if __name__=="__main__":
    TutorialApp().run()
```

Replacing button with label

Lets change the button with a label

```
from kivy.app import App
from kivy.uix.button import Button

class TutorialApp(App):
    def build(self):
        return Label(text='Hello!',font_size=150)

if __name__=="__main__":
    TutorialApp().run()
```

Layouts in Kivy

Anchor layout : kivy.uix.anchorlayout.AnchorLayout

Box layout : kivy.uix.boxlayout.BoxLayout

Float layout : kivy.uix.floatlayout.FloatLayout

Grid layout : kivy.uix.gridlayout.GridLayout

Stack layout : kivy.uix.stacklayout.StackLayout

size_hint property in widget

The size relative to the layout's size instead of an absolute size.

```
widget.size_hint = (width_percent, height_percent)
```

The percent is specified as a floating point number in the range 0-1.

```
widget.size_hint = (0.5, 1.0)
```

If you don't want to use a size_hint for either the width or height, set the value to None. For example, to make a widget that is 250px wide and 30% of the parent's height, do:

```
widget.size_hint = (None, 0.3)
widget.width = 250
```

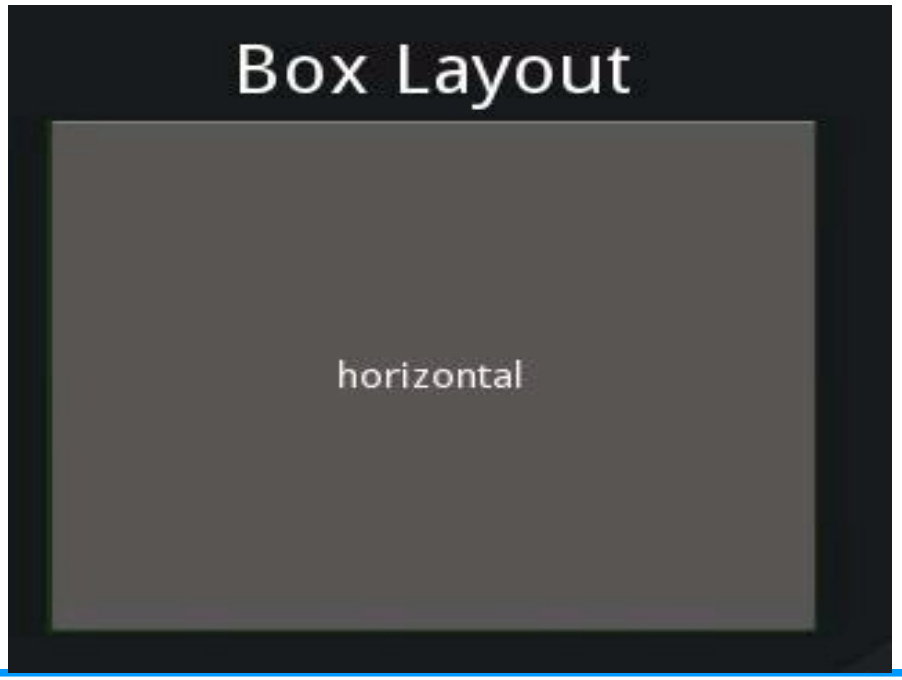
AnchorLayout() kivy.uix.anchorlayout

```
layout = AnchorLayout(
    anchor_x='right', anchor_y='bottom')

btn = Button(text='Hello World')
layout.add_widget(btn)
```

BoxLayout() kivy.uix.boxlayout

```
layout =
BoxLayout(orientation='vertical')
btn1 = Button(text='Hello')
btn2 = Button(text='World')
layout.add_widget(btn1)
layout.add_widget(btn2)
```



With size_hint in Button widget

```
layout = BoxLayout(spacing=10)
btn1 = Button(text='Hello', size_hint=(.7, 1))
btn2 = Button(text='World', size_hint=(.3, 1))
layout.add_widget(btn1)
layout.add_widget(btn2)
```

FloatLayout() kivy.uix.floatlayout

```
layout = FloatLayout(size=(300, 300))
```

Default size_hint=(1, 1), so this button will adopt the same size as the layout:

```
button = Button(text='Hello world')
layout.add_widget(button)
```

To create a button 50% of the width and 25% of the height of the layout and positioned at (20, 20), you can do:

```
button = Button(
    text='Hello world',
    size_hint=(.5, .25),
    pos=(20, 20))
```

If you want to create a button that will always be the size of layout minus 20% on each side:

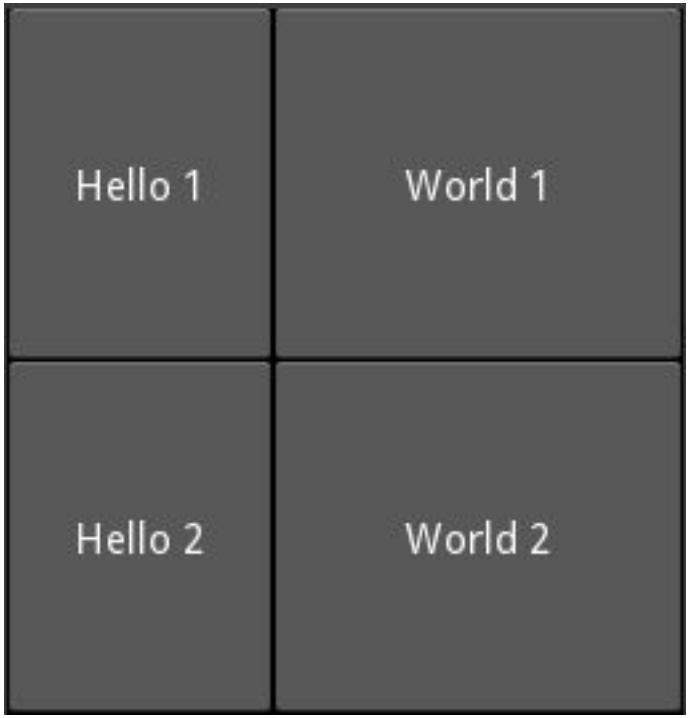
```
button = Button(text='Hello world',
size_hint=(.6, .6),
                pos_hint={'x':.2, 'y':.2})
```

GridLayout() kivy.uix.gridlayout

```
layout = GridLayout(cols=2,rows=2)
```

```
layout = GridLayout(cols=2)
layout.add_widget(Button(text='Hello 1'))
layout.add_widget(Button(text='World 1'))
layout.add_widget(Button(text='Hello 2'))
layout.add_widget(Button(text='World 2'))
```

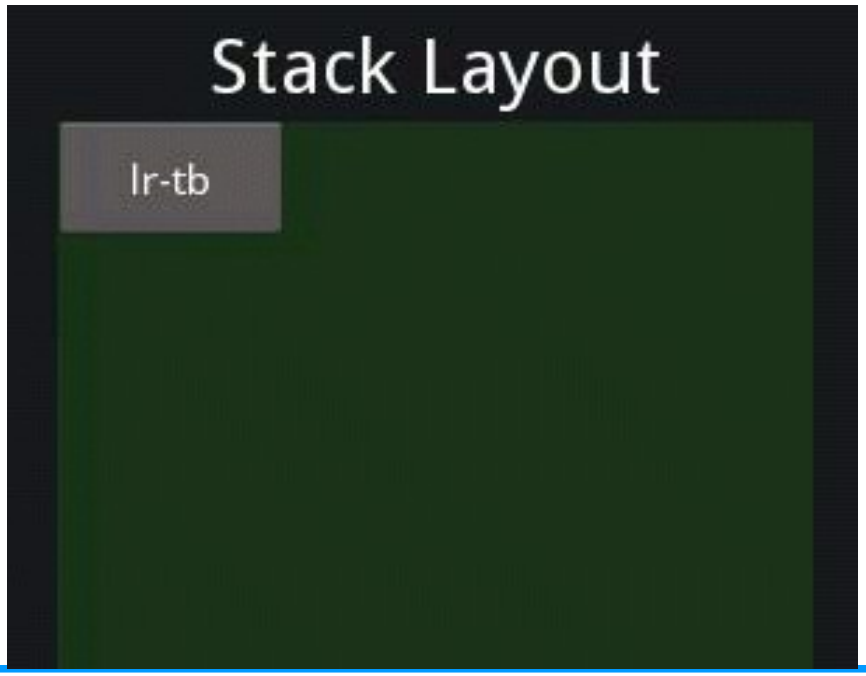
```
layout = GridLayout(cols=2)
layout.add_widget(Button(text='Hello 1',
size_hint_x=None, width=100))
layout.add_widget(Button(text='World 1'))
layout.add_widget(Button(text='Hello 2',
size_hint_x=None, width=100))
layout.add_widget(Button(text='World 2'))
```

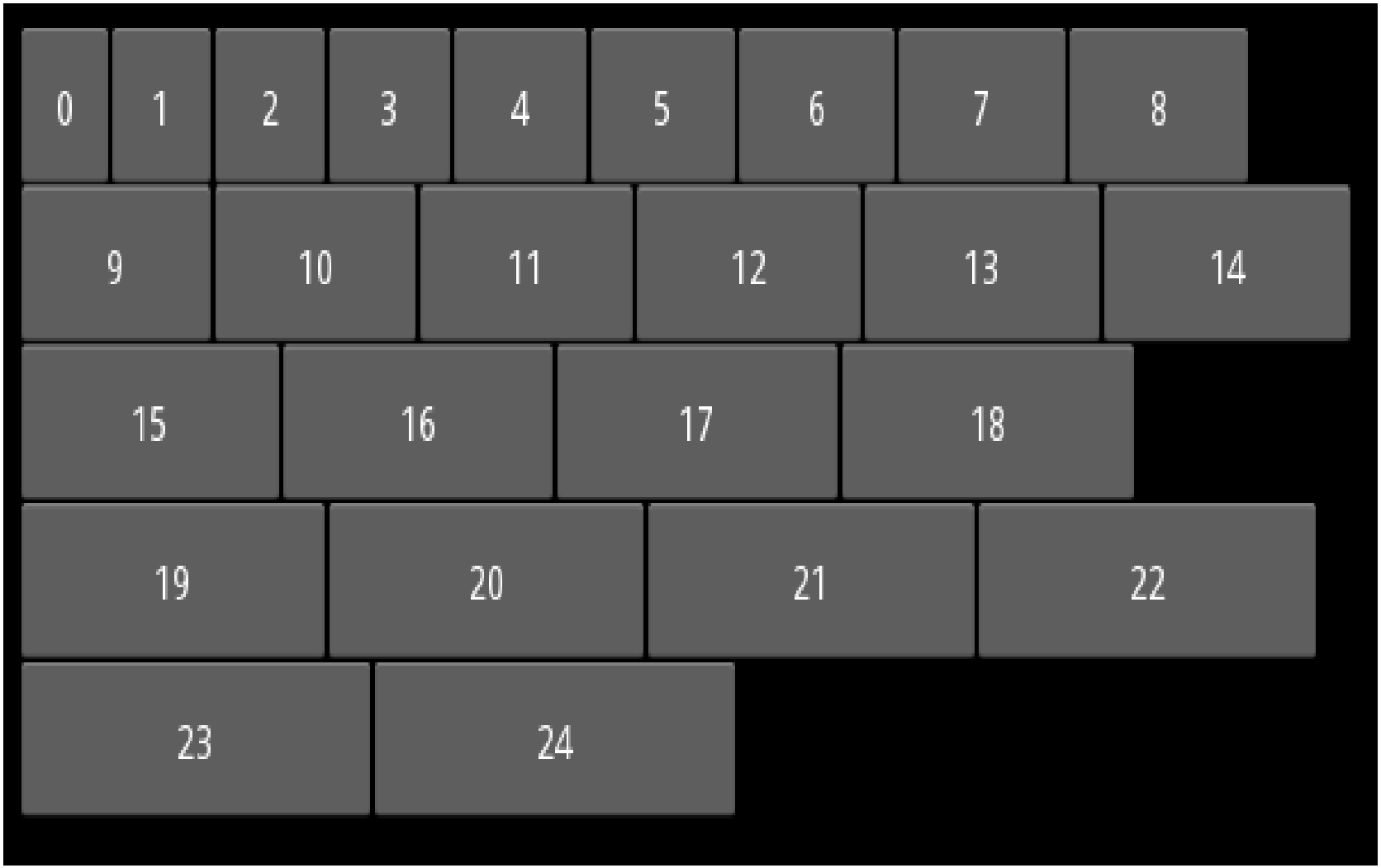


```
layout = GridLayout(cols=2,\
row_force_default=True, row_default_height=40)
layout.add_widget(Button(text='Hello 1',\
size_hint_x=None, width=100))
layout.add_widget(Button(text='World 1'))
layout.add_widget(Button(text='Hello 2',\
size_hint_x=None, width=100))
layout.add_widget(Button(text='World 2'))
```

StackLayout() kivy.uix.stacklayout

```
root = StackLayout()
for i in range(25):
    btn = Button(text=str(i), width=40 + i * 5,
size_hint=(None, 0.15))
    root.add_widget(btn)
```





Lets make the App...