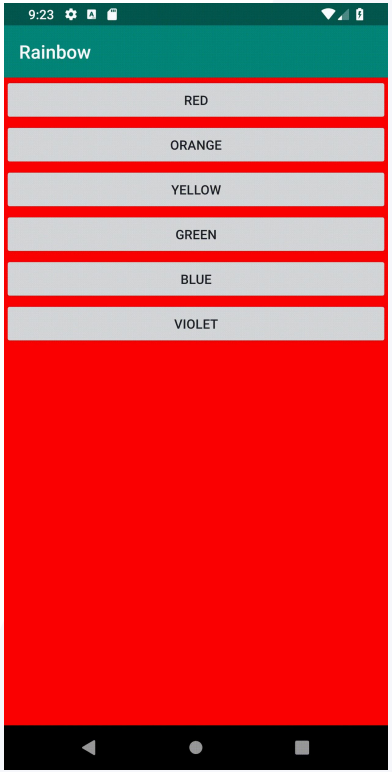
Intro to Android

Upon completion of this module, a student will be able to

- create a project in Android Studio
- prepare a basic testing environment
- edit the user interface using XML
- understand basic application interaction
- use print statements to debug programs
- make code changes that update the application

Project

- Task
  - Build an app with multiple buttons that changes the background to a different color with each button pushed
- Repo
  - https://github.com/LambdaSchool/Android_Rainbow
- Challenge
  - Experiment with different properties of Button and Linear Layout. To improve the look of your app.



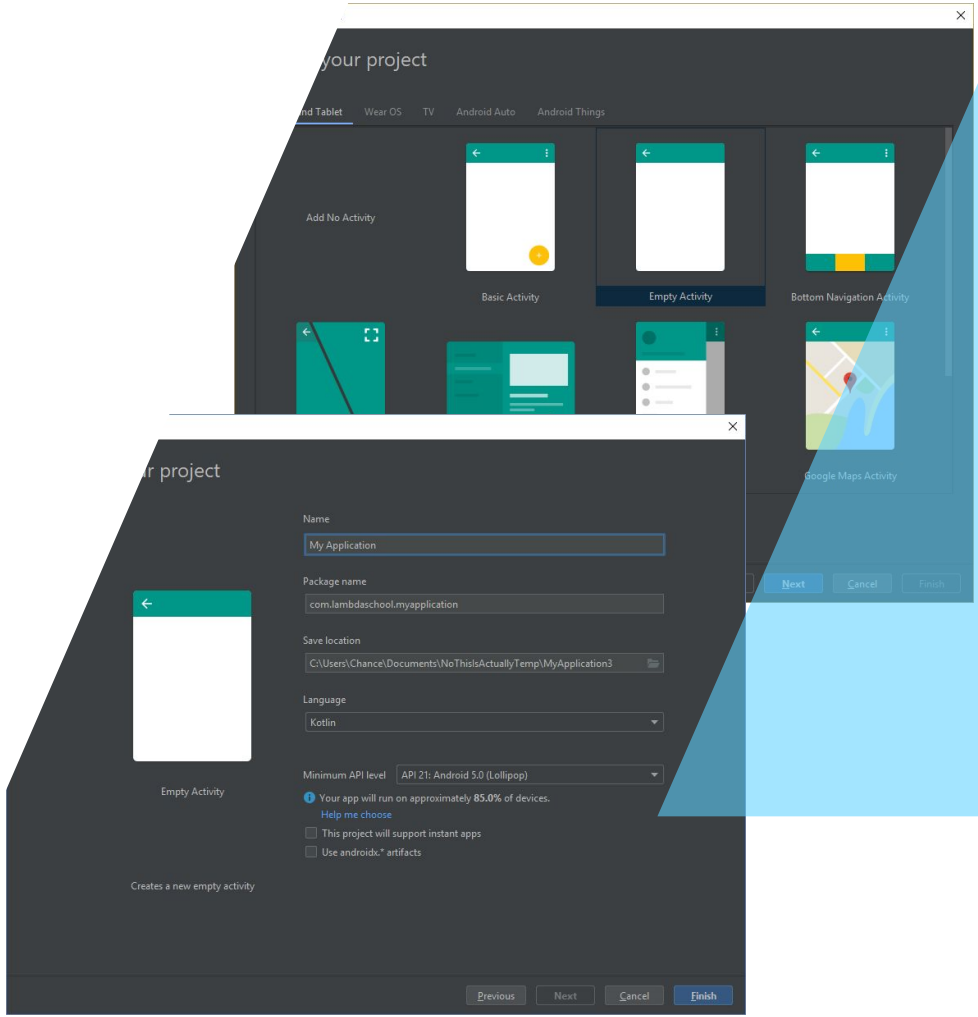

Lambda
A Student Can
create a project in Android Studio

Android Studio

- IDE
  - Code Editor
  - Build Automation Tools
  - Debugger
- Download and manage Android libraries
- Create and Run Android Virtual Devices (AVDs)

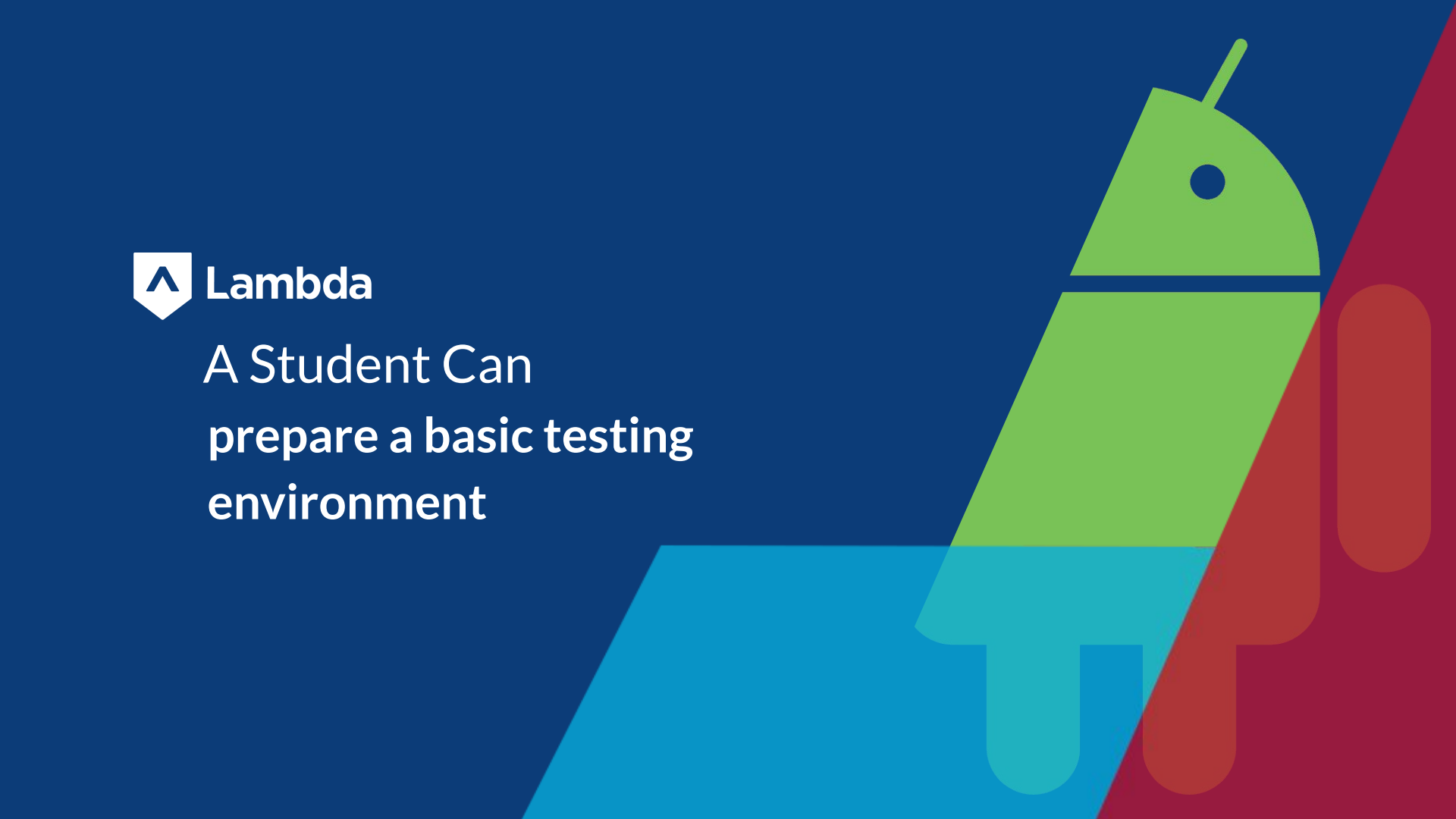
Intro with Image

- Create Android Project
- Select Activity Type
- Name the App
- Target Android Devices



Challenge

- Explore the files generated by our new project


Lambda
A Student Can
prepare a basic testing
environment

Android Virtual Device

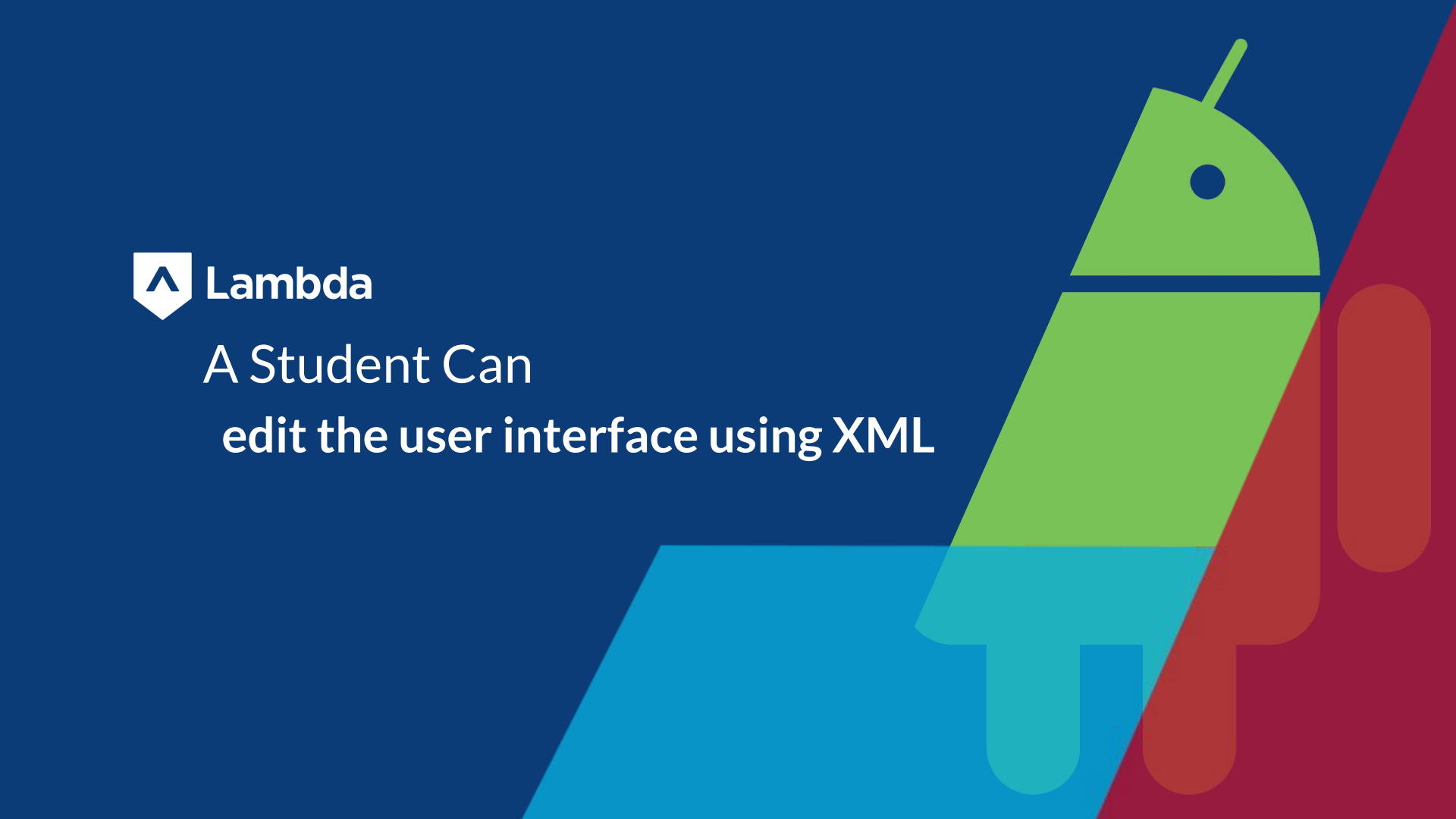
- Great tool for quickly testing apps
- Can build based on a variety of devices
- Requires a relatively powerful computer to run smoothly
- Built and managed with AVD Manager

Physical Android Device

- Run on Actual Hardware
- Test devices from different manufacturers
- Devices with different idiosyncrasies

Challenge

- Experiment with the Virtual device and see how it feels to work with


Lambda
A Student Can
edit the user interface using XML

UI Object

- Filing Cabinet
  - Multiple Drawers
    - Label
    - Handles
    - Attributes
- Attaching Process
  - Give the component an id
  - Use the id to get a handle in the Kotlin code

XML

- View type (drawer)
- Attribute (file folder)
- Value (data in file)

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

    <TextView
            android:id="@+id/my_text_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World!" />

</LinearLayout>
```

Challenge

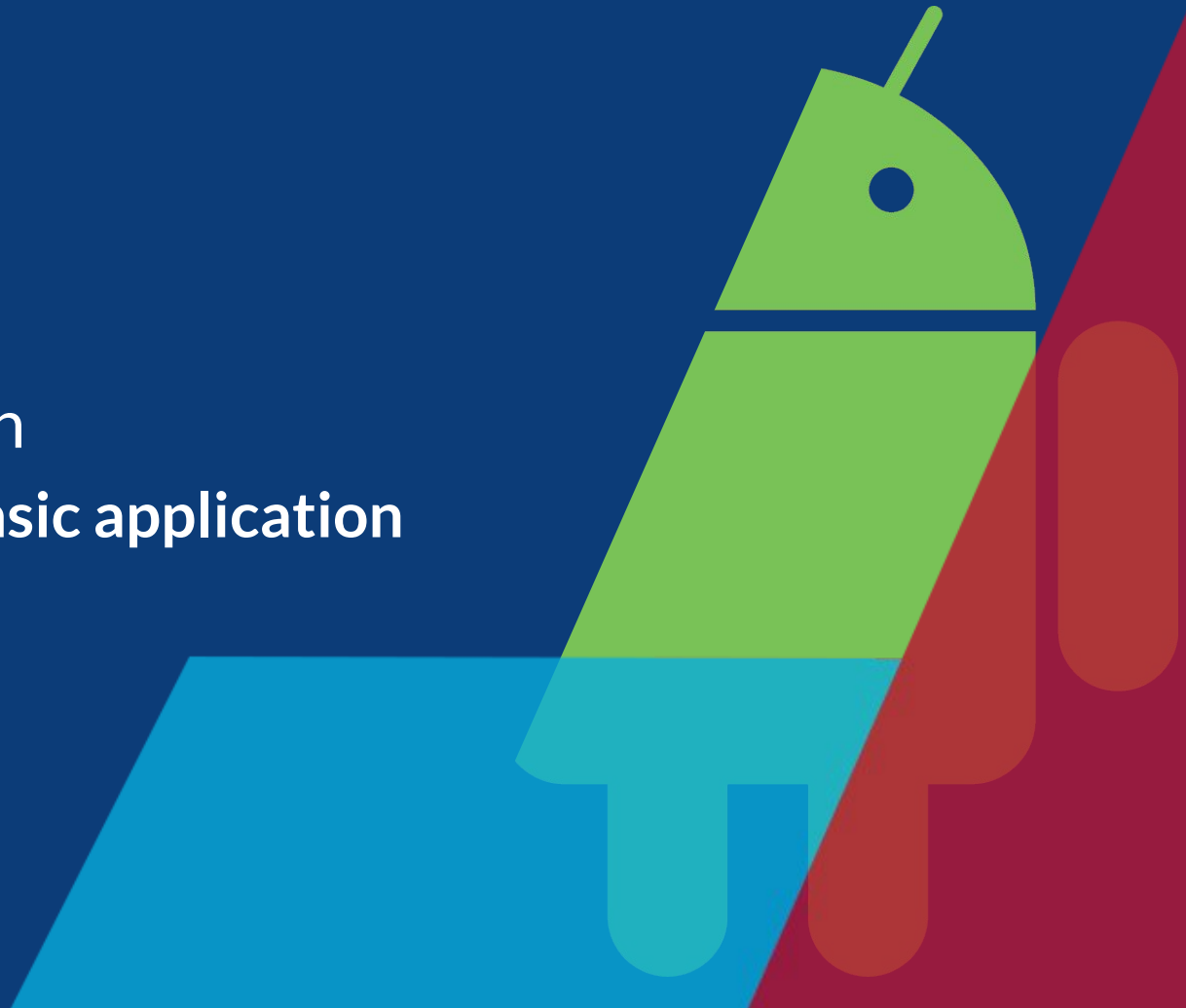
- Adjust the android:Text attribute in the xml to get the textview to say what you want

Solution

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

    ...

    <TextView
            android:id="@+id/my_text_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="This is my value" />

</LinearLayout>
```

A Student Can

understand basic application interaction



Listener

- Listener is assigned
- Message is sent
- Code is executed

On Click Listener

- Listener
- Responds to `Click` interaction

```
my_button.setOnClickListener
{
    // task to be performed
}
```

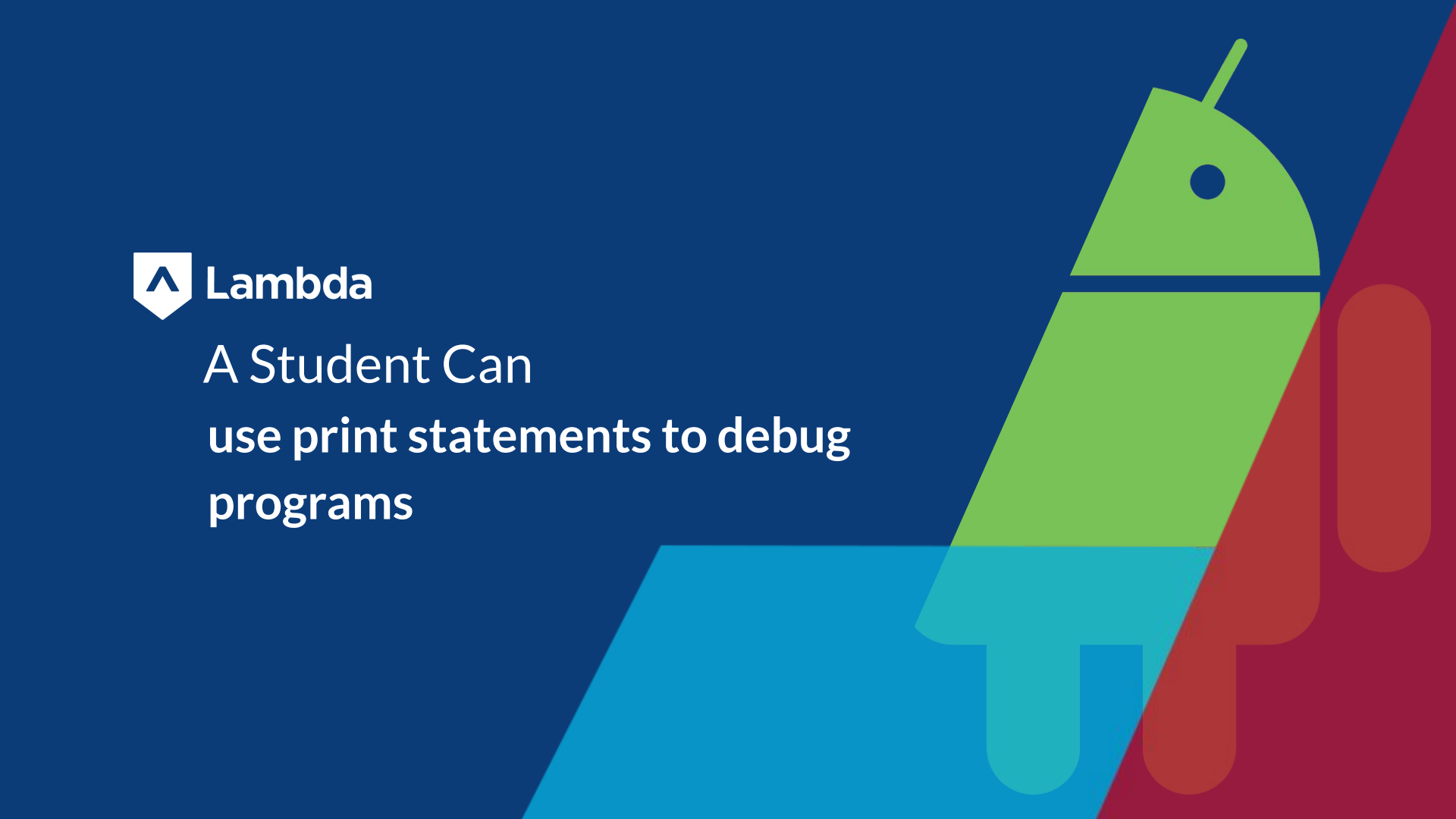
Challenge

1. Add an id to one of the buttons in your layout (xml)
2. Add a listener to the button (kotlin)

Solution

```
class MainActivity : AppCompatActivity() {

   override fun onCreate(...) {
       super.onCreate(savedInstanceState)

setContentView(R.layout. activity_main)

       button_right.setOnClickListener  {  }
   }
}
```

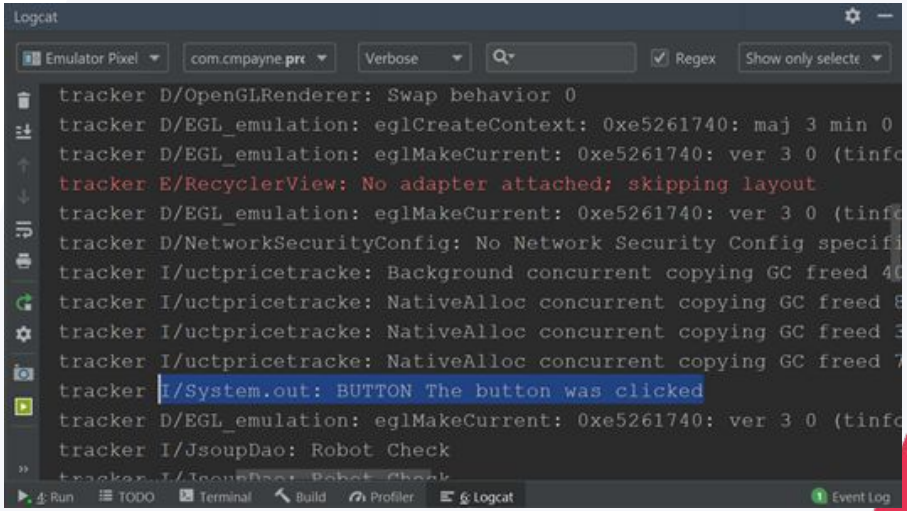
```
<LinearLayout
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       tools:context=".MainActivity" >

   ...

   <Button
           android:id="@+id/button_right"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="Button" />

</LinearLayout>
```


Lambda
A Student Can
use print statements to debug programs

Put one Foot in Front of the Other

Print Statements

```
purchase_button.setOnClickListener {
    println("BUTTON was clicked")
    // task to be performed
}
```



Challenge

- Put a print statement inside of your listener

Solution

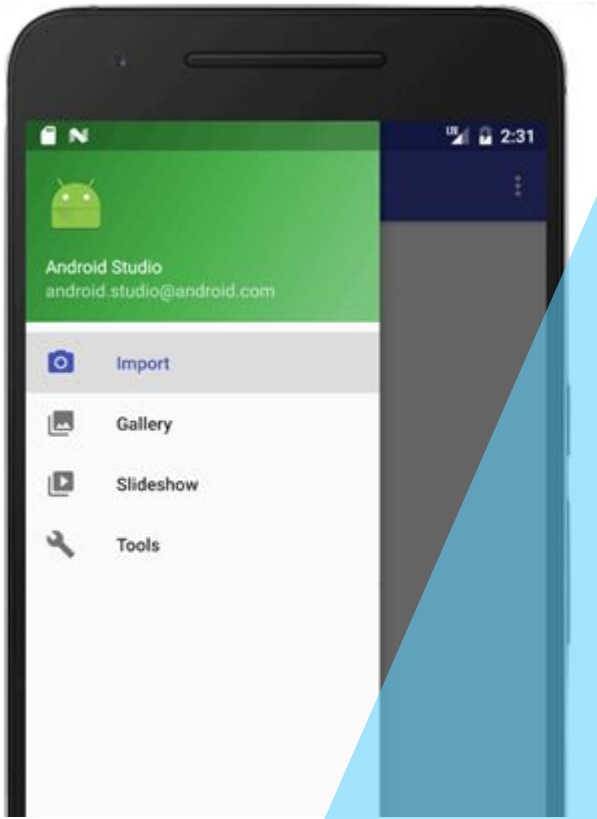
```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout. activity_main)
        ...
        button_right.setOnClickListener  {
            println("BUTTON right was clicked" )
        }
    }
}
```

A Student Can

make code changes that update the application

Updating the UI

- Update Information to the User
- Prompt additional interaction
- Programmatically change component attributes



Putting it All Together

- Add an ID another component
- Get a handle to that component
- Change that component

Challenge

1. Add an id to your Text View
2. In your listener, us the “text” property to update the text for your textview

> The setText call should look like:

```
text_id.text = "my text"
```

Solution

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout. activity_main)
        ...
        button_right.setOnClickListener  {
            println("BUTTON right was clicked" )
            text_id.text = "my text"
        }
    }
}
```