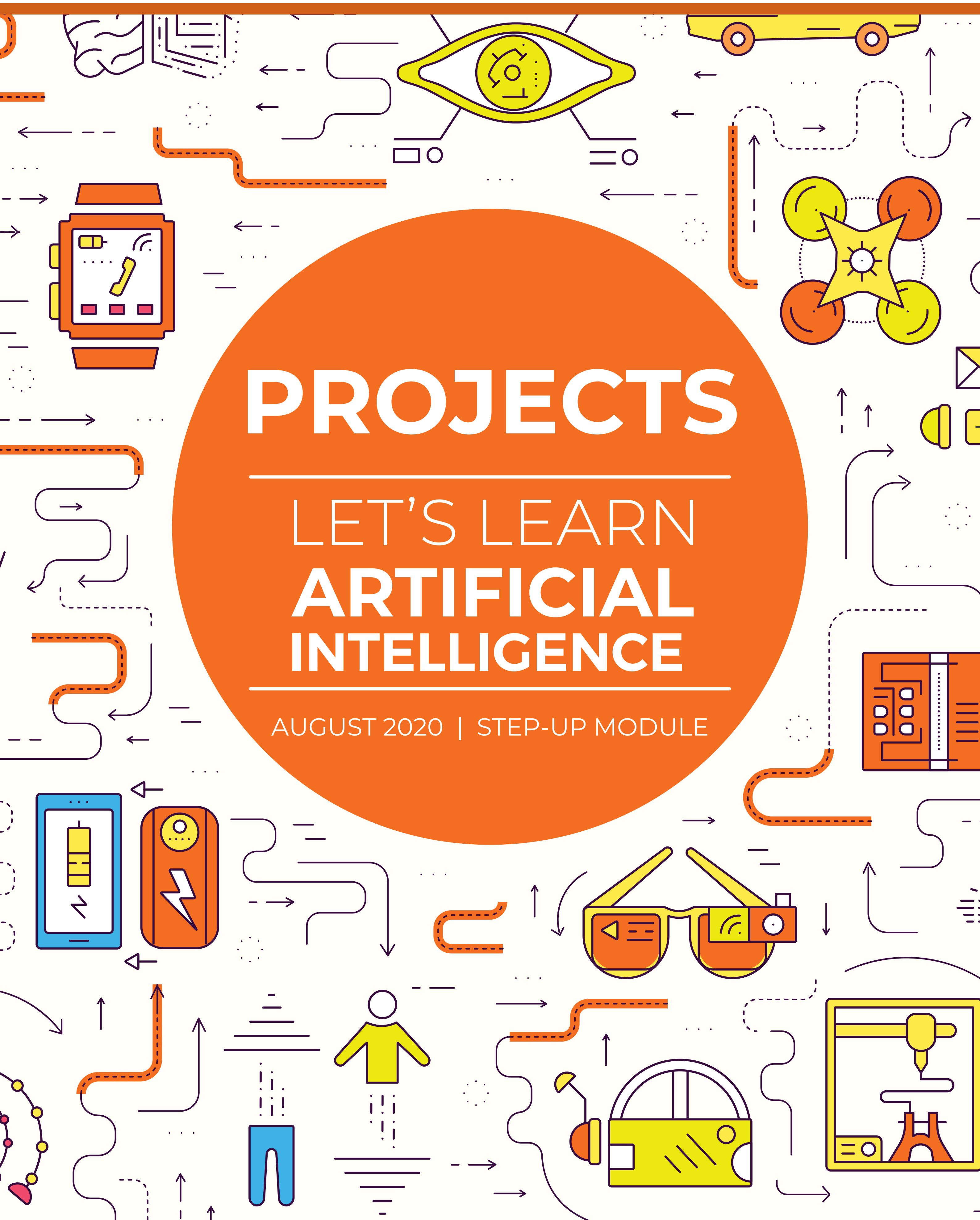
NITI Aayog | ATAL INNOVATION MISSION

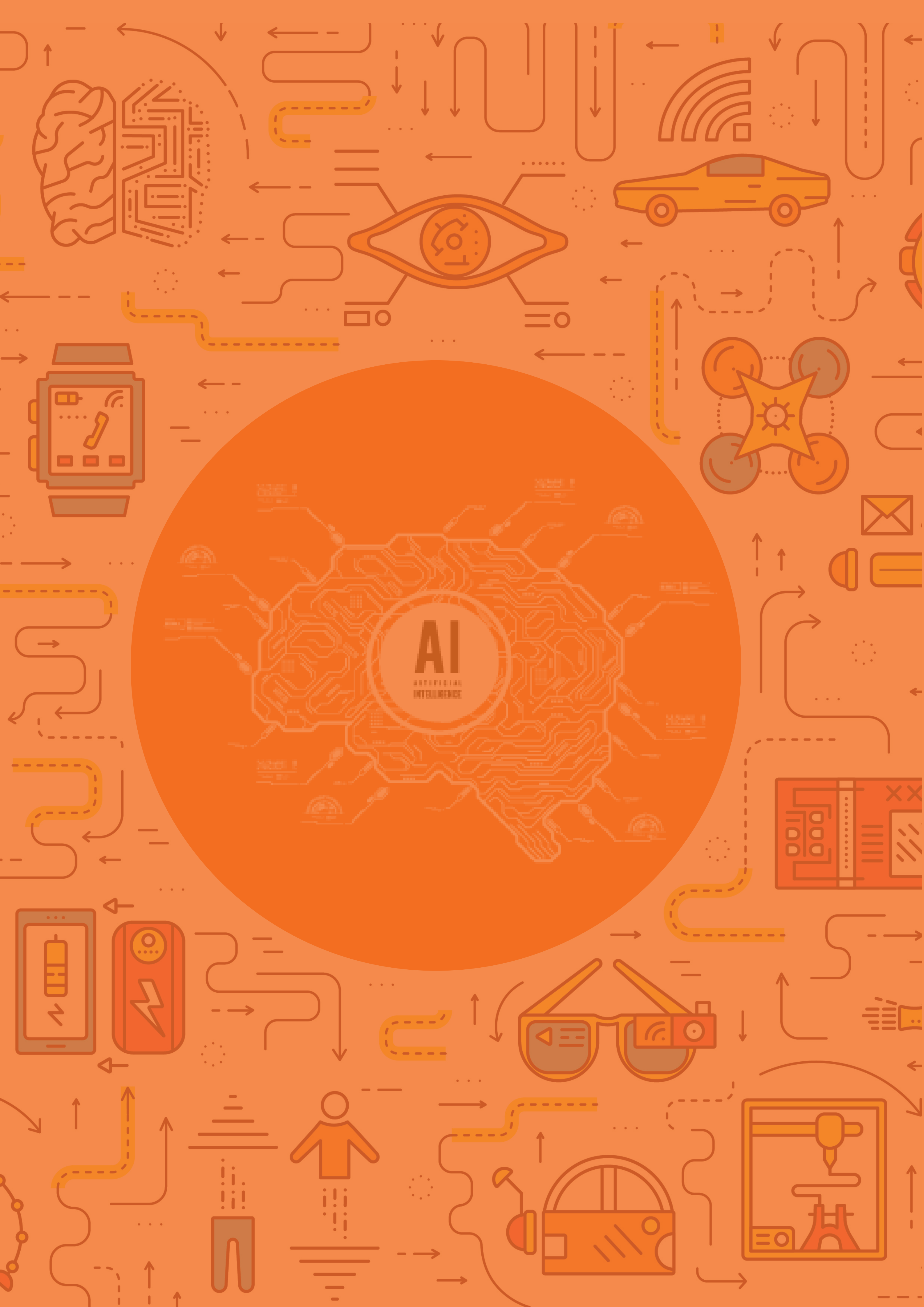


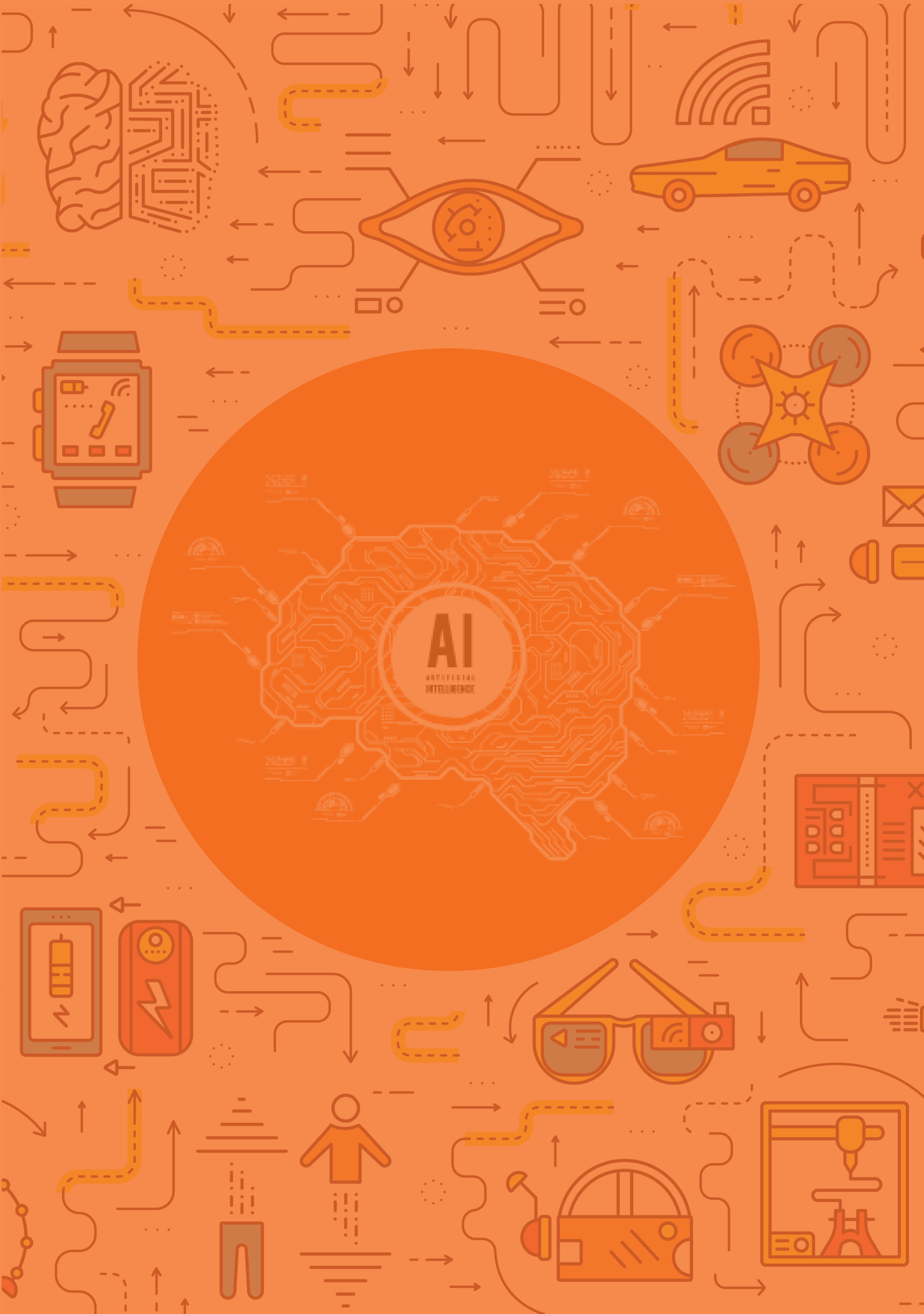
PROJECTS

LET'S LEARN

ARTIFICIAL INTELLIGENCE

AUGUST 2020 | STEP-UP MODULE


AI
ARTIFICIAL
INTELLIGENCE


AI
ARTIFICIAL
INTELLIGENCE

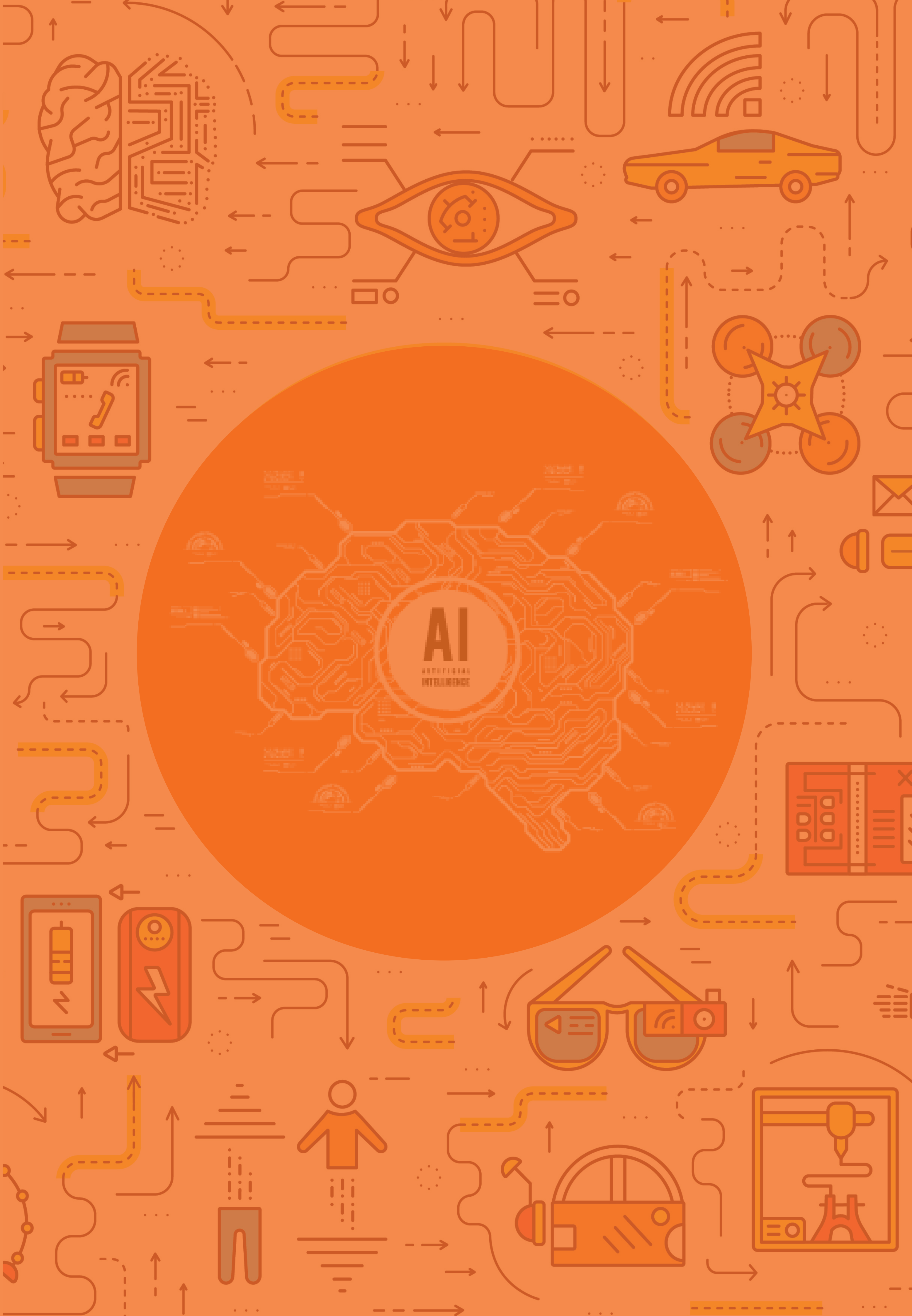
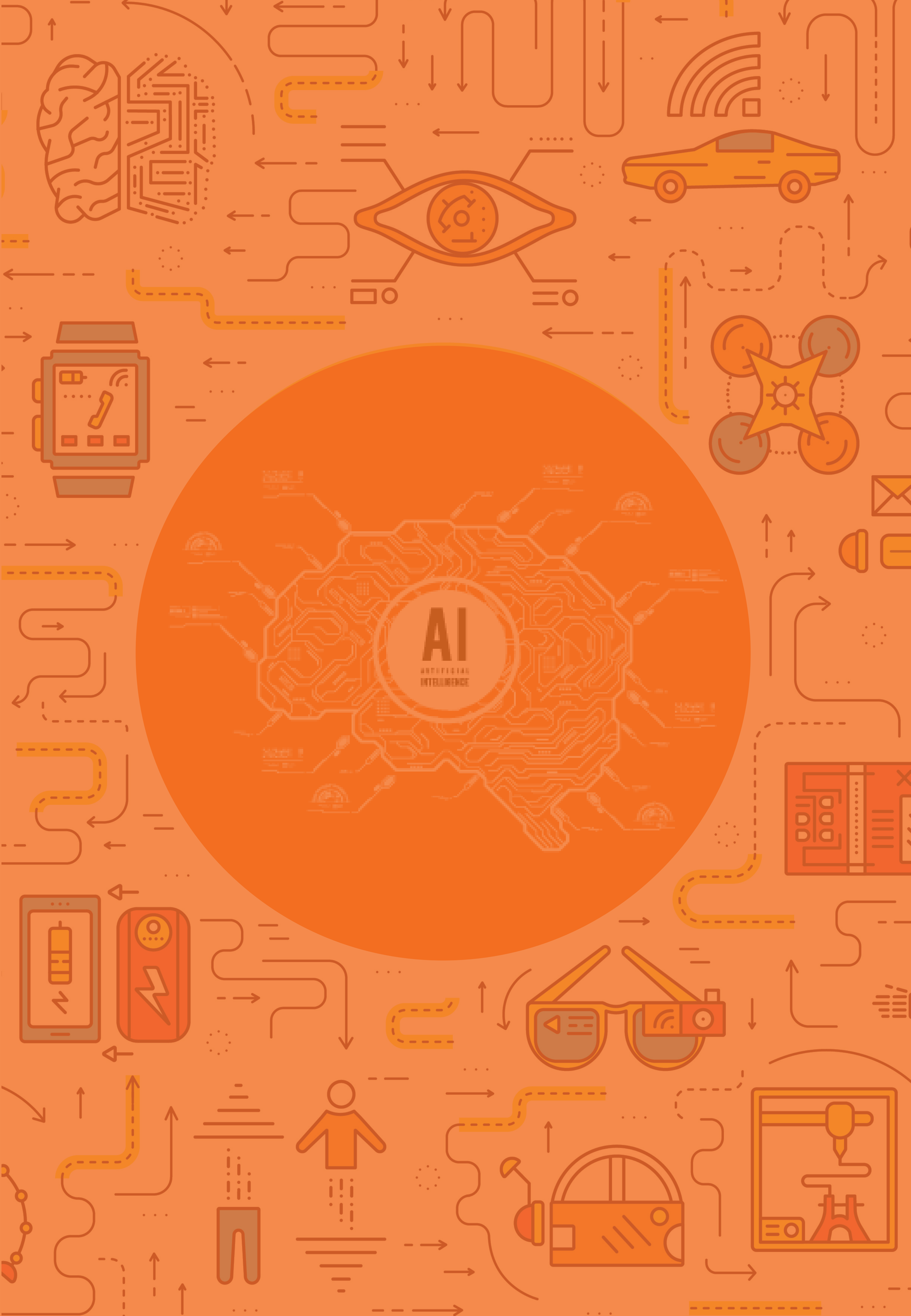

AI
ARTIFICIAL
INTELLIGENCE

TABLE OF CONTENTS


AI
ARTIFICIAL
INTELLIGENCE

ARTIFICIAL LEARNING
PROJECTS

Key Learning Outcomes

At the end of this module, you will be able to:

a) Perform artificial intelligence tasks using Scratch, APIs and Python
b) Build simple artificial intelligence solutions
c) Build artificial intelligence solutions for real-world problems using Python

Topic Covered

Foundation Projects for AI | Advanced Projects for AI

TOPIC 1 - FOUNDATIONAL AI PROJECTS

1.1ROCK PAPER SCISSORS GAME

In this project you will train a machine to identify your hand gesture and play a rock paper scissor game against a computer.

PROJECT DESCRIPTION

Rock paper scissors is a hand game in which each player simultaneously forms one of three moves with an outstretched hand. These moves are 'rock", "paper", and "scissors" as shown in the image.

Each move wins against one shape and loses to another.

- Rock 'crushes' scissors but is 'covered' by paper
- Paper 'covers' rock but is 'cut' by scissors
- Scissors is 'crushed' by rock but 'cuts' paper

PREREQUISITES

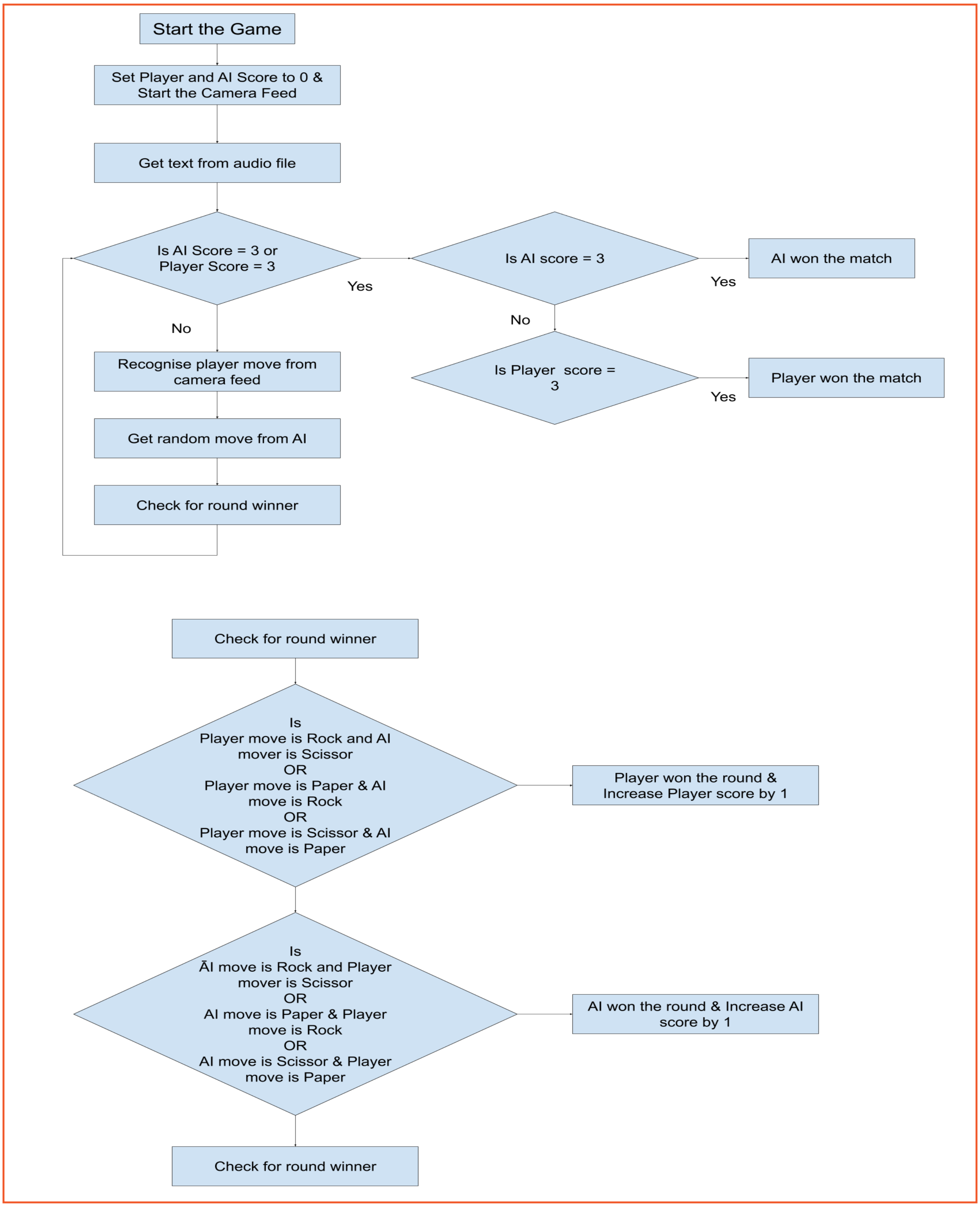
Compatible platforms:

1. Machine Learning for Kids
2. Cognimates
3. PictoBlox

THE SOLUTION

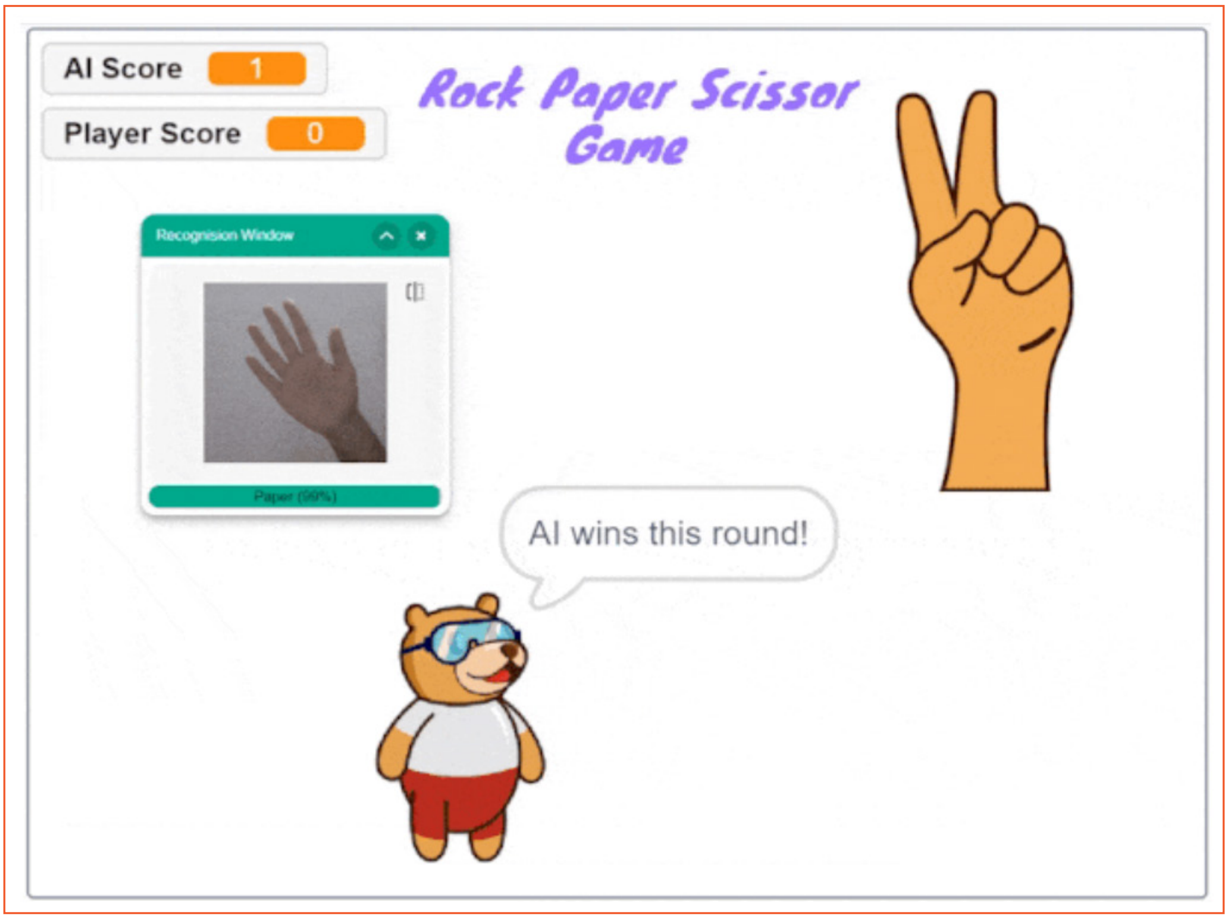
The Flow Chart

The following flow chart shows the overview of the program required to implement Rock Paper Scissors Game in the compatible platforms. You can find various graphical blocks in the compatible platforms to make the game. Go ahead!


Start the Game
Set Player and AI Score to 0 & Start the Camera Feed
Get text from audio file
Is AI Score = 3 or Player Score = 3
Yes
No
Is AI score = 3
Yes
AI won the match
No
Is Player score = 3
Yes
Player won the match
Recognise player move from camera feed
Get random move from AI
Check for round winner
Check for round winner
Is Player move is Rock and AI mover is Scissor OR Player move is Paper & AI move is Rock OR Player move is Scissor & AI move is Paper
Player won the round & Increase Player score by 1
Is AI move is Rock and Player mover is Scissor OR AI move is Paper & Player move is Rock OR AI move is Scissor & Player move is Paper
AI won the round & Increase AI score by 1
Check for round winner

Task:

Your task is to create this project using Scratch. train an image model that can detect hand signs of Rock, Paper, and Scissors. Once the model is trained, create a script using that. Set the initial scores to zero and then detect the player's move using the camera/webcam. Next, randomly generate the machine's move and match it with the player's move to find out the result and accordingly assign the points. Whoever from the two i.e. the player or the computer scores 3 points first, will win the match.



EXPLORE

Find out other common games that you can create projects for. For those who want to explore the usage of Keras and OpenCV use the following link to understand how you can create this game with these.

https://www.youtube.com/watch?time_continue=36&v=0uSA3xyXlwM&feature=emb_logo

1.2 ML WITH HARDWARE

So far, you have made many interesting projects using AI and ML, but it's not just about software or apps. We can control robots and many other hardware related things as well with ai and ml. Let's make a gesture controlled robot using arduino or micro:bit.

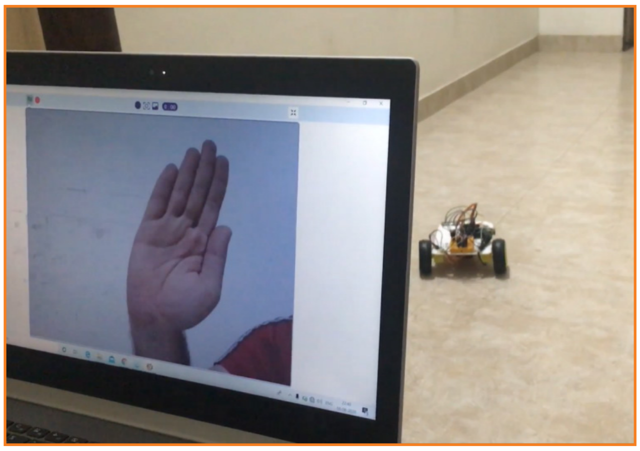
PROJECT DESCRIPTION

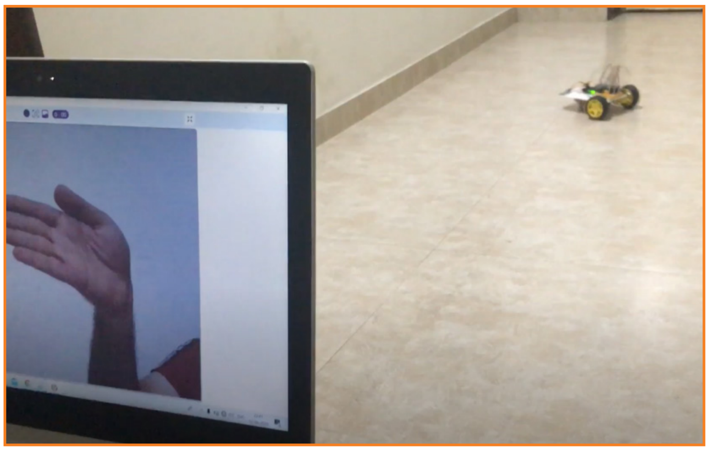
You have to control a robot by identifying different hand gestures from the camera.

You made mobile robots and have controlled it using your keyboard, smartphone, joystick and what not. But, this time let's level up and control the 2-Wheel Drive Robot using the hand gestures.

The controls are:

1. If we keep our palm straight, the robot should move forward.
2. If we tilt our hand in the left direction, the robot should turn left.
3. If we tilt our hand in the right direction, the robot should turn right.
4. If we close the fist, the robot should stop moving.





Learning Outcomes:

1. Physical interaction of ML
2. Controlling robot using Bluetooth

PREREQUISITES

1. Arduino Uno OR micro:bit
2. SN7544one motor driver
3. 7805 Power IC
4. HC-05 bluetooth module,
5. 2 (or 4) Dual Shaft Motor
6. 2 (or 4) Wheels
7. Spacers
8. Castor Wheel,
9. 2 Cell Battery
10. Battery Holder
11. Jumper Cables
12. Nuts and Bolts for mounting
13. Laptop with a good internet connection.

Compatible platforms

1. PictoBlox (for arduino or microbit)
2. Makecode (for microbit)
3. Scratch 3.0 (for microbit)

ASSEMBLY & CIRCUITRY

Option 1 - Making the Robot using Arduino

EXPLORE

To assemble the robot, kindly follow the steps given here:

Sample assemble and circuitry of a robot using Arduino:

https://bit.ly/Make4WheelDriveRobot

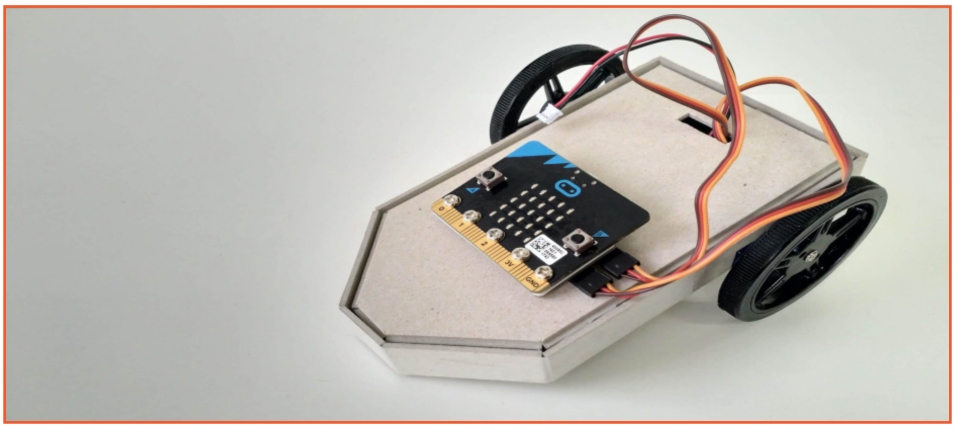
Option 2 - Making the Robot using Micro:bit

EXPLORE

To assemble the micro:bit robot, kindly follow the steps given here:

Sample assemble and circuitry of a robot using Micro:bit:

https://tinkercademy.com/tutorials/krazy-kar-v2/



TRAINING THE MACHINE LEARNING MODEL

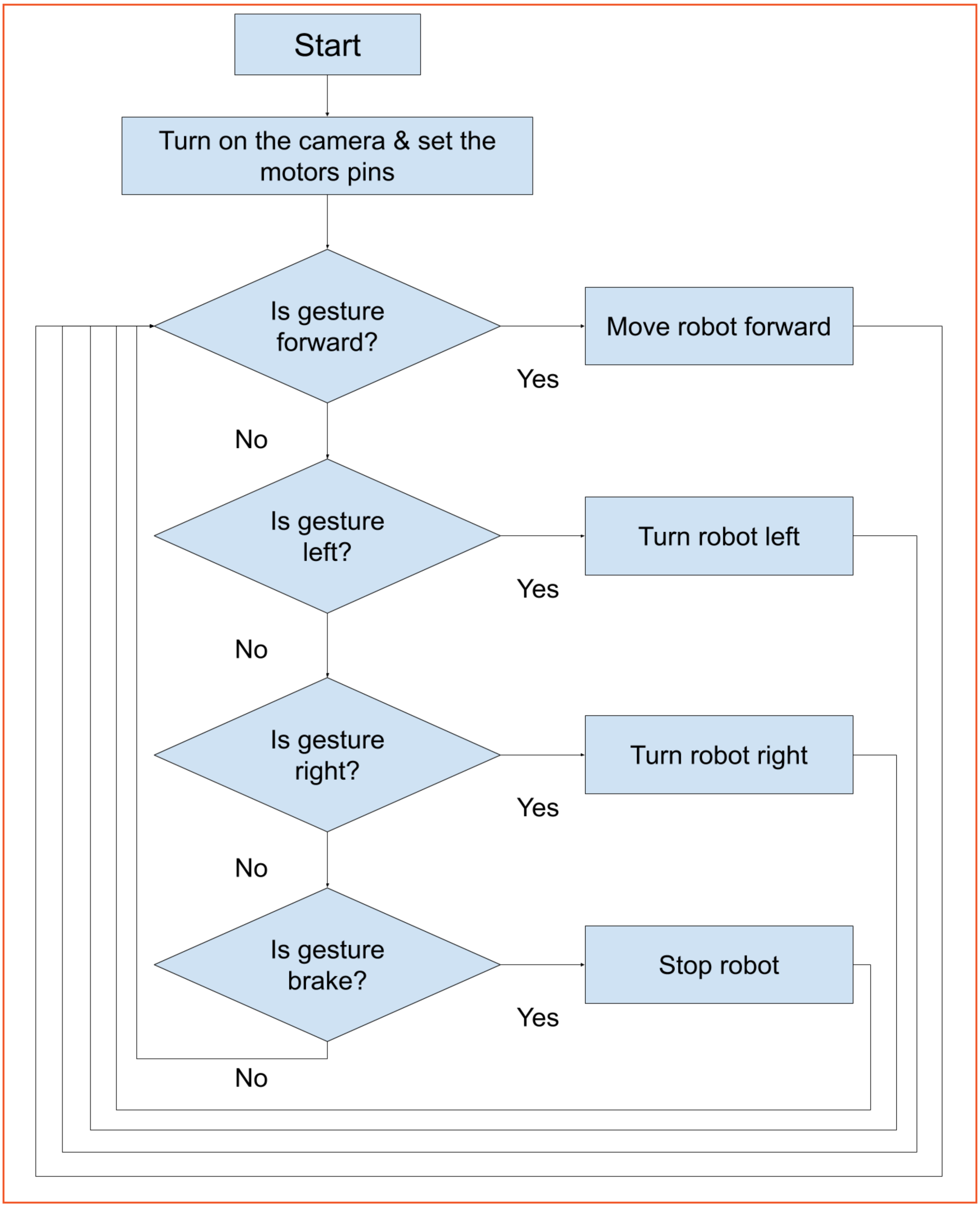
You will need to train the model with four classes for turning the robot towards the right, left, moving straight and stop it with four unique hand gestures like turning the palm towards right, left, keeping straight and fist closed respectively as shown in the starting of the project using a teachable machine

EXPLORE

(https://teachablemachine.withgoogle.com).

FLOW CHART

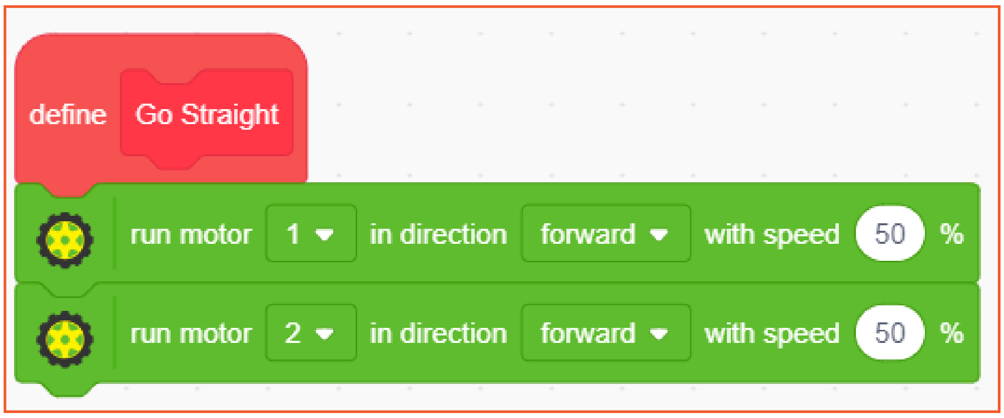
The following flow chart shows the overview of the program required to implement Gesture controlled Robot using Arduino or Micro:Bit in the compatible platforms. You can find various graphical blocks in the compatible platforms to make the face expression recognizer. Go ahead!


Start
Turn on the camera & set the motors pins
Is gesture forward?
Yes
Move robot forward
No
Is gesture left?
Yes
Turn robot left
No
Is gesture right?
Yes
Turn robot right
No
Is gesture brake?
Yes
Stop robot
No

Option 1 - Code for Arduino Robot

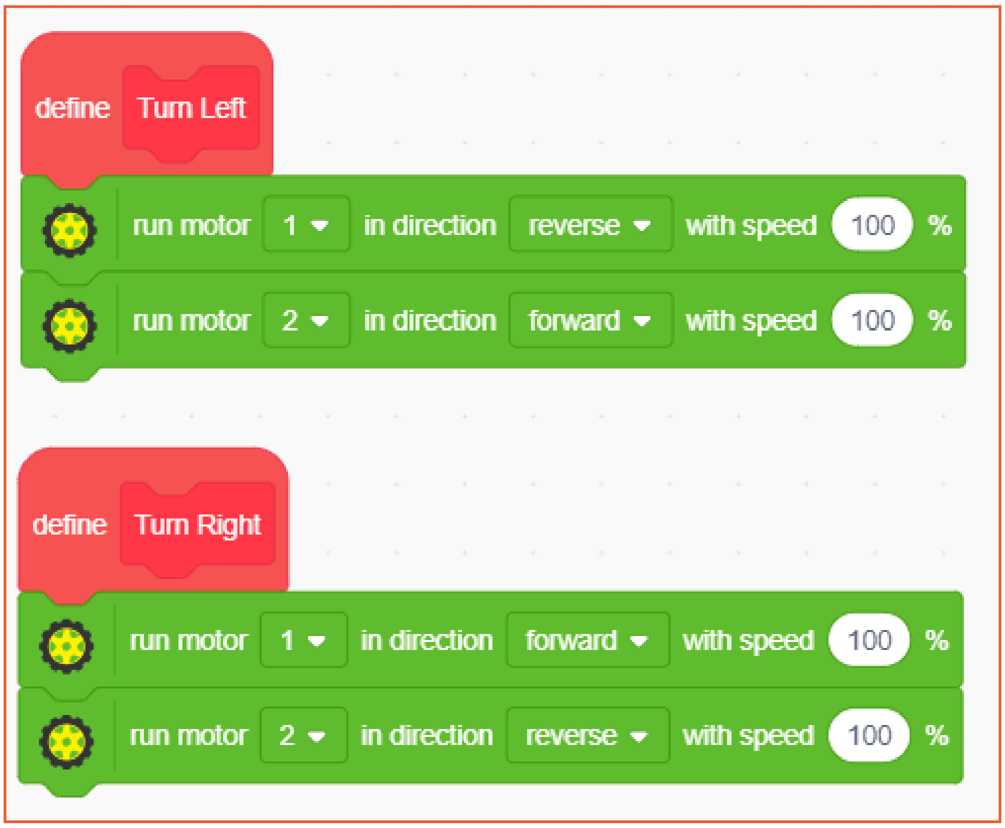
The following code snippets show a sample code of PictoBlox for making the Gesture controlled Arduino Robot.

1. Blocks for moving straight:



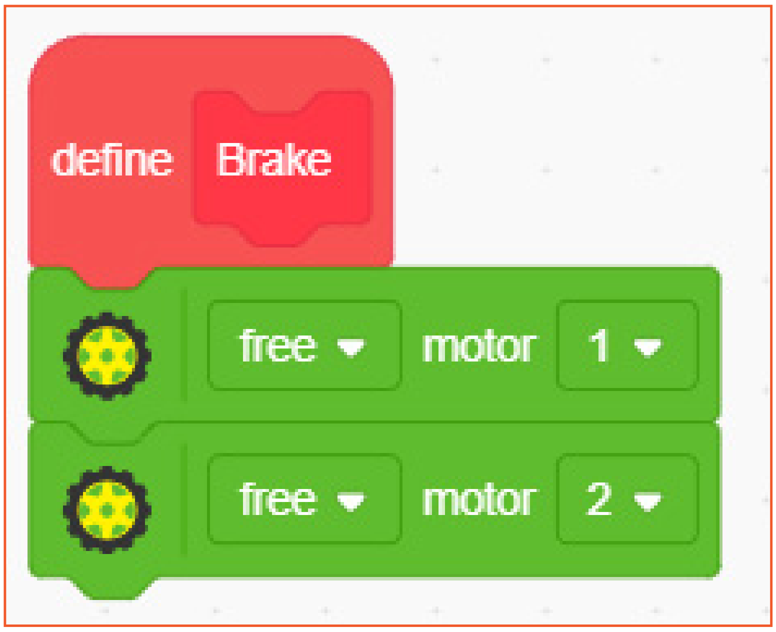
```
define Go Straight
run motor 1 in direction forward with speed 50 %
run motor 2 in direction forward with speed 50 %
```

2. Blocks for turning left and right:



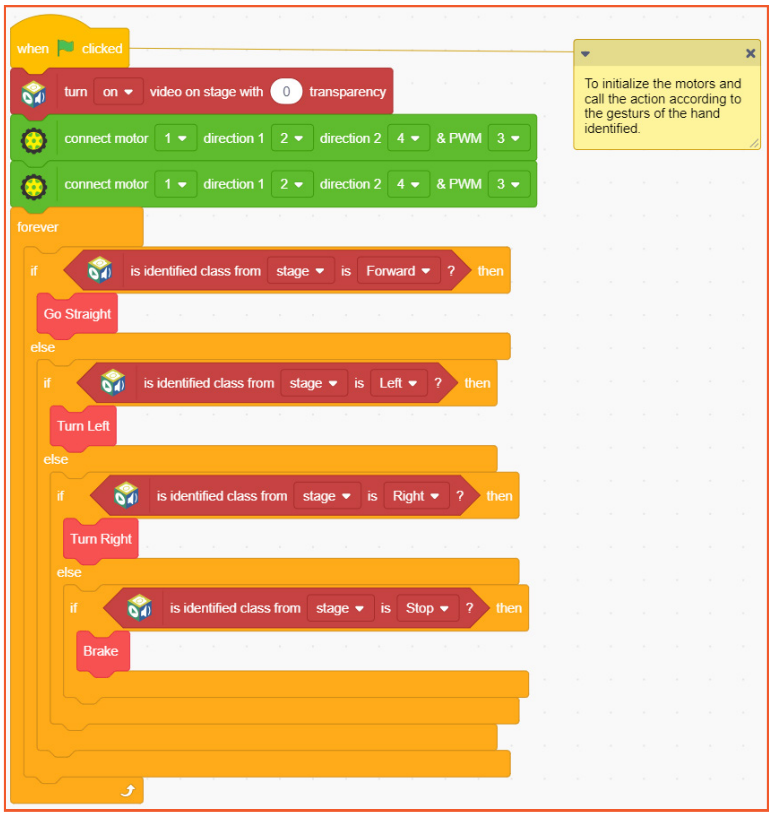
```
define Turn Left
run motor 1 in direction reverse with speed 100 %
run motor 2 in direction forward with speed 100 %

define Turn Right
run motor 1 in direction forward with speed 100 %
run motor 2 in direction reverse with speed 100 %
```

3. Blocks for stopping the robot:



```
define Brake
free ▾ motor 1 ▾
free ▾ motor 2 ▾
```

4. Main code to control the robot:



```
when 🏳 clicked
turn on ▾ video on stage with 0 transparency
connect motor 1 ▾ direction 1 2 ▾ direction 2 4 ▾ & PWM 3 ▾
connect motor 1 ▾ direction 1 2 ▾ direction 2 4 ▾ & PWM 3 ▾
forever
  if is identified class from stage ▾ is Forward ▾ ? then
    Go Straight
  else
    if is identified class from stage ▾ is Left ▾ ? then
      Turn Left
    else
      if is identified class from stage ▾ is Right ▾ ? then
        Turn Right
      else
        if is identified class from stage ▾ is Stop ▾ ? then
          Brake
```

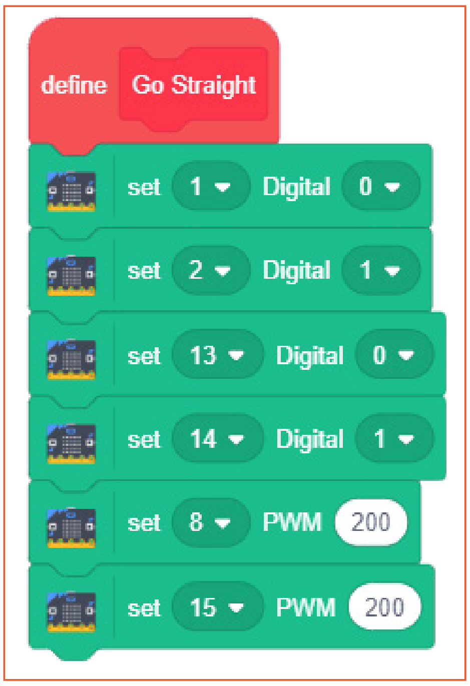
To initialize the motors and call the action according to the gesturs of the hand identified.

This project shows how we can even control robots with gestures using Machine Learning.

Option 2 - Code for Micro:bit Robot

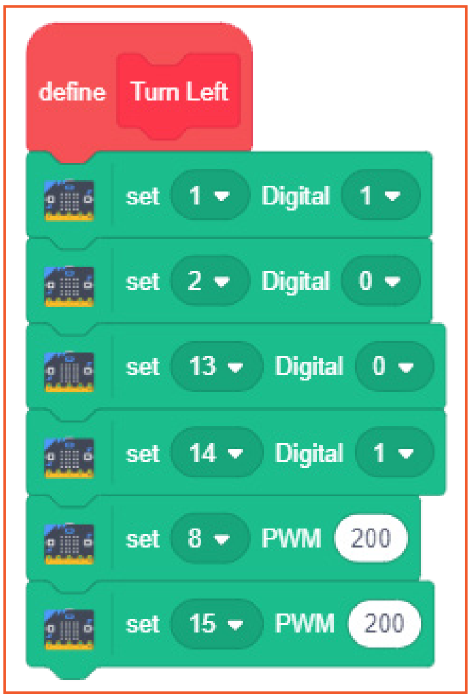
Following is the sample code for MicroBit Robot:

1. Blocks for moving straight:



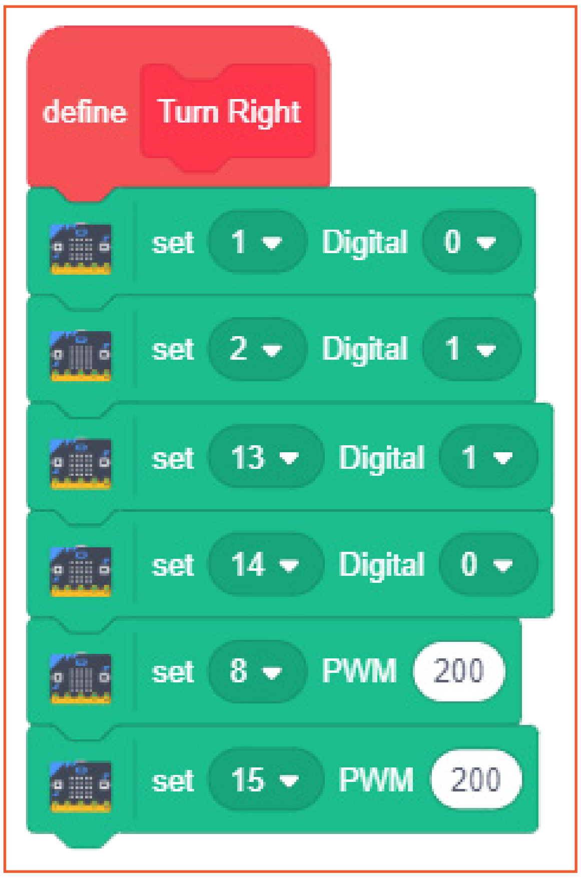
```
define Go Straight
set 1 Digital 0
set 2 Digital 1
set 13 Digital 0
set 14 Digital 1
set 8 PWM 200
set 15 PWM 200
```

2. Blocks for turning left:



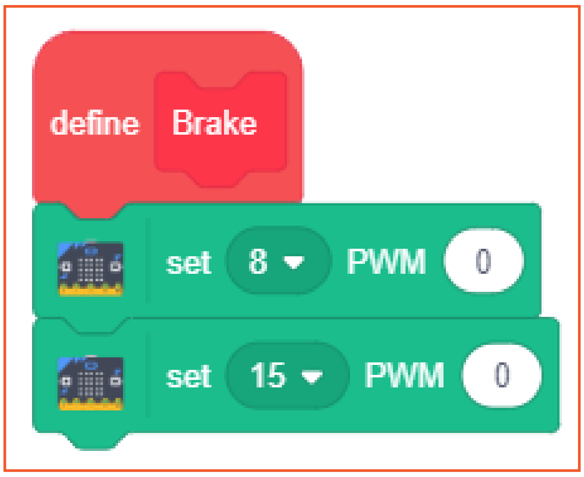
```
define Turn Left
set 1 Digital 1
set 2 Digital 0
set 13 Digital 0
set 14 Digital 1
set 8 PWM 200
set 15 PWM 200
```

3. Blocks for turning right:



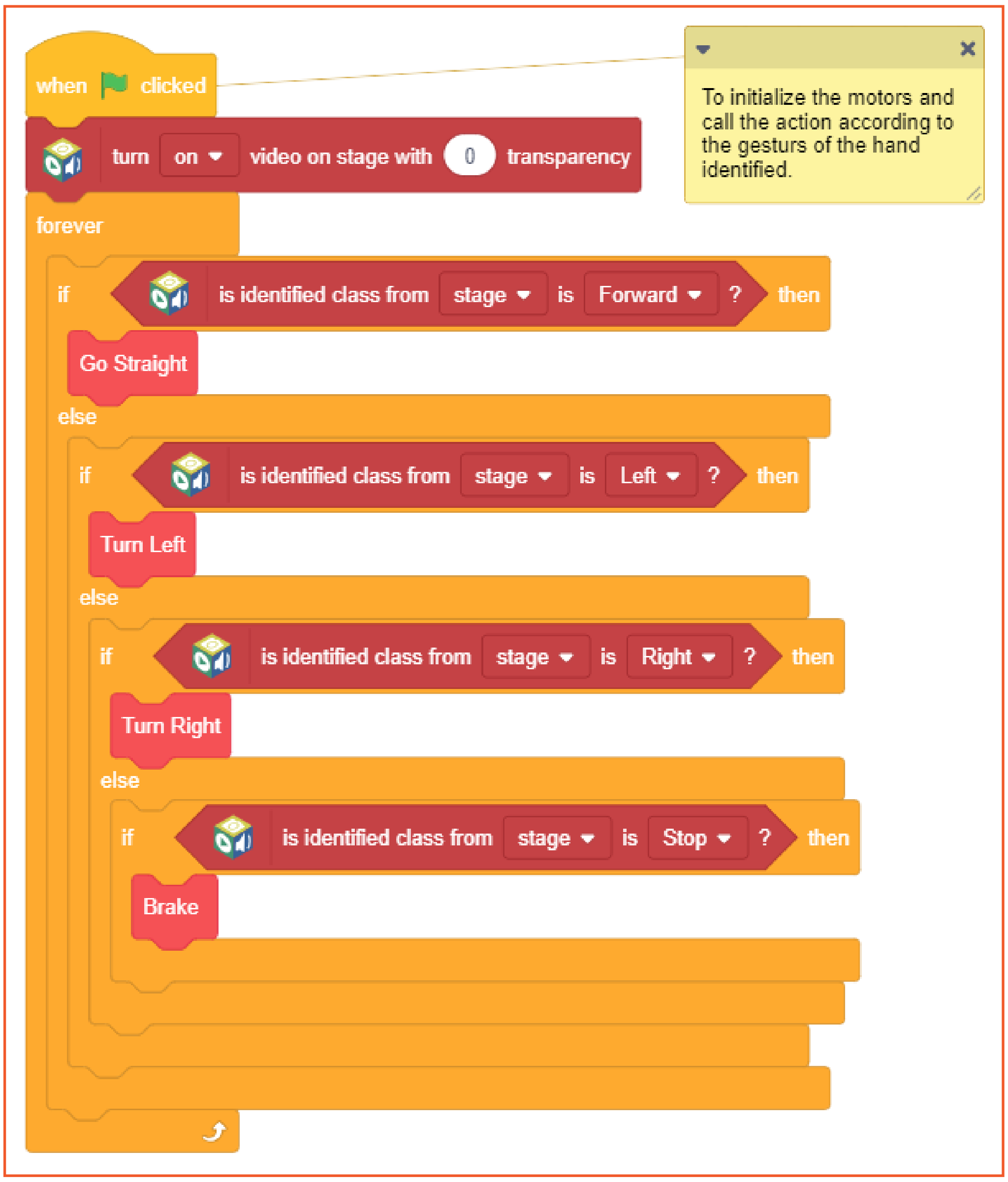
```
define Turn Right
set 1 Digital 0
set 2 Digital 1
set 13 Digital 1
set 14 Digital 0
set 8 PWM 200
set 15 PWM 200
```

4. Blocks for stopping the robot:



```
define Brake
set 8 PWM 0
set 15 PWM 0
```

5. Main code to control the robot:



```
when [flag] clicked
turn (on ▾) video on stage with (0) transparency
forever
    if < is identified class from (stage ▾) is (Forward ▾) ? > then
        Go Straight
    else
        if < is identified class from (stage ▾) is (Left ▾) ? > then
            Turn Left
        else
            if < is identified class from (stage ▾) is (Right ▾) ? > then
                Turn Right
            else
                if < is identified class from (stage ▾) is (Stop ▾) ? > then
                    Brake
```

To initialize the motors and call the action according to the gesturs of the hand identified.

HOW DOES IT WORK?

Let's understand the logic behind it i.e. how it works.

First we train an image model that can detect hand signs/gestures. Once the model is trained and exported, we will create a script using that. In the beginning, we need to define the various pins of the motor driver and then we will continuously detect the class from the stage where webcam/camera feed is available. The robot will perform certain actions i.e. moving forward, turning left or right, or getting stopped as per the class detected.

EXPLORE MORE

We challenge you to create an AI based robot that can solve a real-world problem. It can be a healthcare warrior or an agribot or a modern warfare robot. Brainstorm on various ideas and come up with exciting ideas.

EXPLORE

You can explore option 1 more at:

https://bit.ly/ArduinoRobotAI

EXPLORE

You can explore option 2 more at:

https://tinkercademy.com/tutorials/krazy-kar-v2/

1.3 CONVERSATIONAL AI INTERFACE

A chatbot is an AI tool that allows users to type their questions, make orders, and solve problems. Users type their questions or requests for the AI tool, and the AI tool responds similarly to a human by providing the request or asking for additional information. We are going to use AWS to build an AI chatbot in the cloud. AWS has a service called Amazon Lex that allows us to develop conversational chatbots using Amazon Alexa technology.

THE PROBLEM

A new pizza restaurant is opening up in town. They want to make it simple for customers to place orders. They feel the easier it is to order the pizza, the more likely customers will be to return to the restaurant. You will take on the role of AI designer and design a chatbot for customers to use to order pizza.

GOALS OF THE PROJECT

At the end of the project, the student will be able to

1. create a custom chatbot for users to order a pizza and an advertisementusing Amazon Lex to create the bot and the Amazon Management Console to test the bot.

PRE-REQUISITE

Signup in aws educate.

Step:1

EXPLORE

Go to:

https://www.awseducate.com/Registration

Choose “Student” as your role and click “Next”

Step 2: Enter your information into the fields. Click the icons on the right for more detailed

information. Enter your country of residence, your first name and your last name (surname).

Enter your institution (school) issued email address: e.g. St u dent@amazonu.edu .

If you do not have an institution issued email address, additional verification may be required. You will receive further instructions in your email.

Enter your birth month and year, your institution name, your expected graduation, or program completion date. If you have a valid promo code* you can apply it in the bottom field. After completing the CAPTCHA, click "Next".

Step 3: Please review AWS Educate's Terms and Conditions. Selecting "Agree" allows you to proceed.

Step 4: Check your email to follow the automated email verification process.

Step 5: After email verification, you will be prompted to set a password to log into the student portal. Make sure your password meets the required security level.

Now you are able to log in!

(The AWS Educate Starter Account (ESA) offers you free access to AWS cloud resources without requiring a credit card for payment. With an ESA, you will receive $100 in preloaded credits at member institutions, or $30 in preloaded credits at non-member institutions.Your AWS Educate Starter Account will renew on an annual basis as long as you are an active student and a member of AWS Educate. Account credit is automatically added to your AWS Educate Starter Account once every 12 months from the date your application is approved until your graduation/ program completion date.)

TUTORIALS TO LEARN ABOUT BUILDING CHATBOTS

EXPLORE

Watch this video introducing Amazon Lex:

https://www.youtube.com/watch?v=d3LYlNqfuzI&feature=youtu.be.

THE SOLUTION

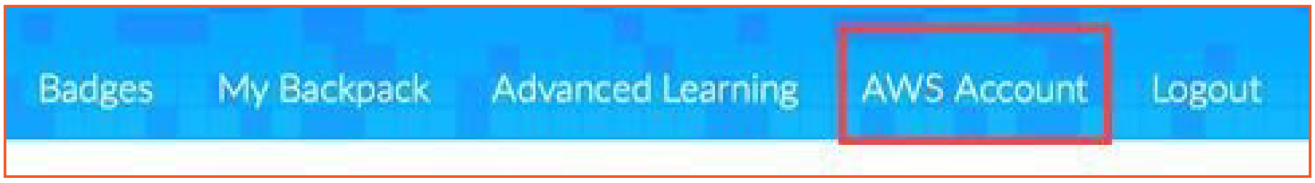
How to proceed towards a solution:

Let's start building our chatbot! Follow along closely with the instructions. We will walk through the steps of getting set up together, and then you'll have the chance to build your chatbot on your own.

Project the steps on the screen using the teacher computer. Model each step for the students using the teacher computer and projector.

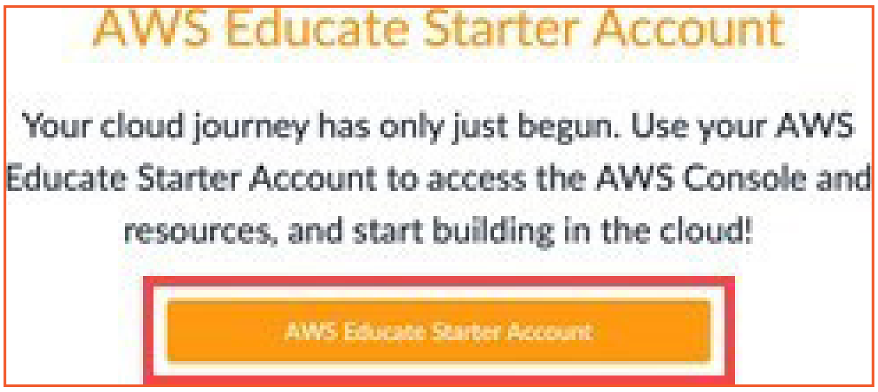
Step 1. Log in to AWS Educate.

- Go to awseducate.com.
- Log in to your AWS Educate starter account.
- Click Login to AWS Educate and sign in using your AWS Educate account.
- Students should have an AWS Educate login from the "Introduction to AWS Educate" module. If not, please review that activity and create an account.
- Once logged in to the AWS Educate portal, select AWS Account.

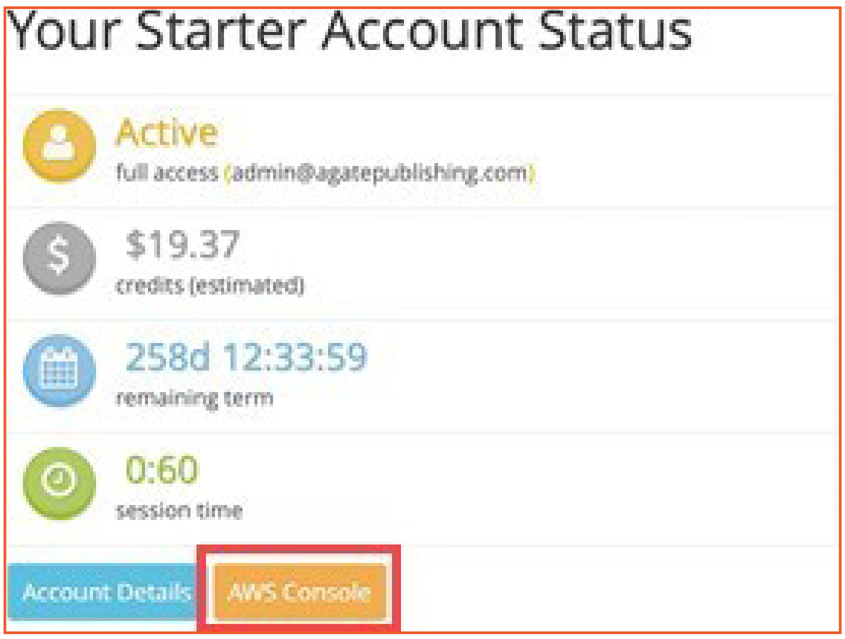


Step 2. On the new screen, select AWS Educate Starter Account in the center of the screen

Step 3. On the next new screen, click on AWS Console.


AWS Educate Starter Account

Your cloud journey has only just begun. Use your AWS Educate Starter Account to access the AWS Console and resources, and start building in the cloud!



Your Starter Account Status

Active
full access (admin@agatepublishing.com)

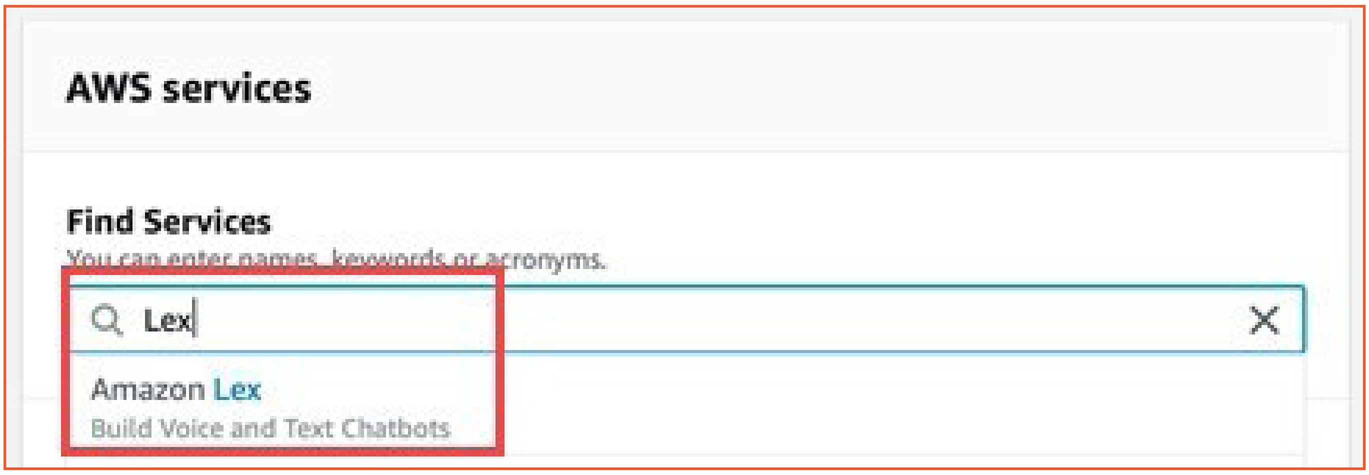
$19.37
credits (estimated)

258d 12:33:59
remaining term

0:60
session time

The AWS Management Console will now be displayed. This is a menu for all cloud services AWS provides.

Step 4. Navigate to Amazon Lex. Click the search bar and search for and then click Amazon Lex.



AWS services

Find Services
You can enter names, keywords or acronyms.

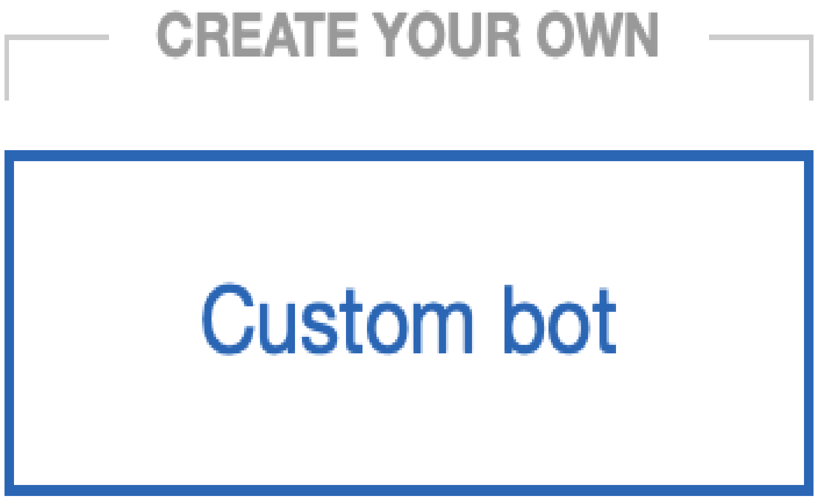
Lex

Amazon Lex
Build Voice and Text Chatbots

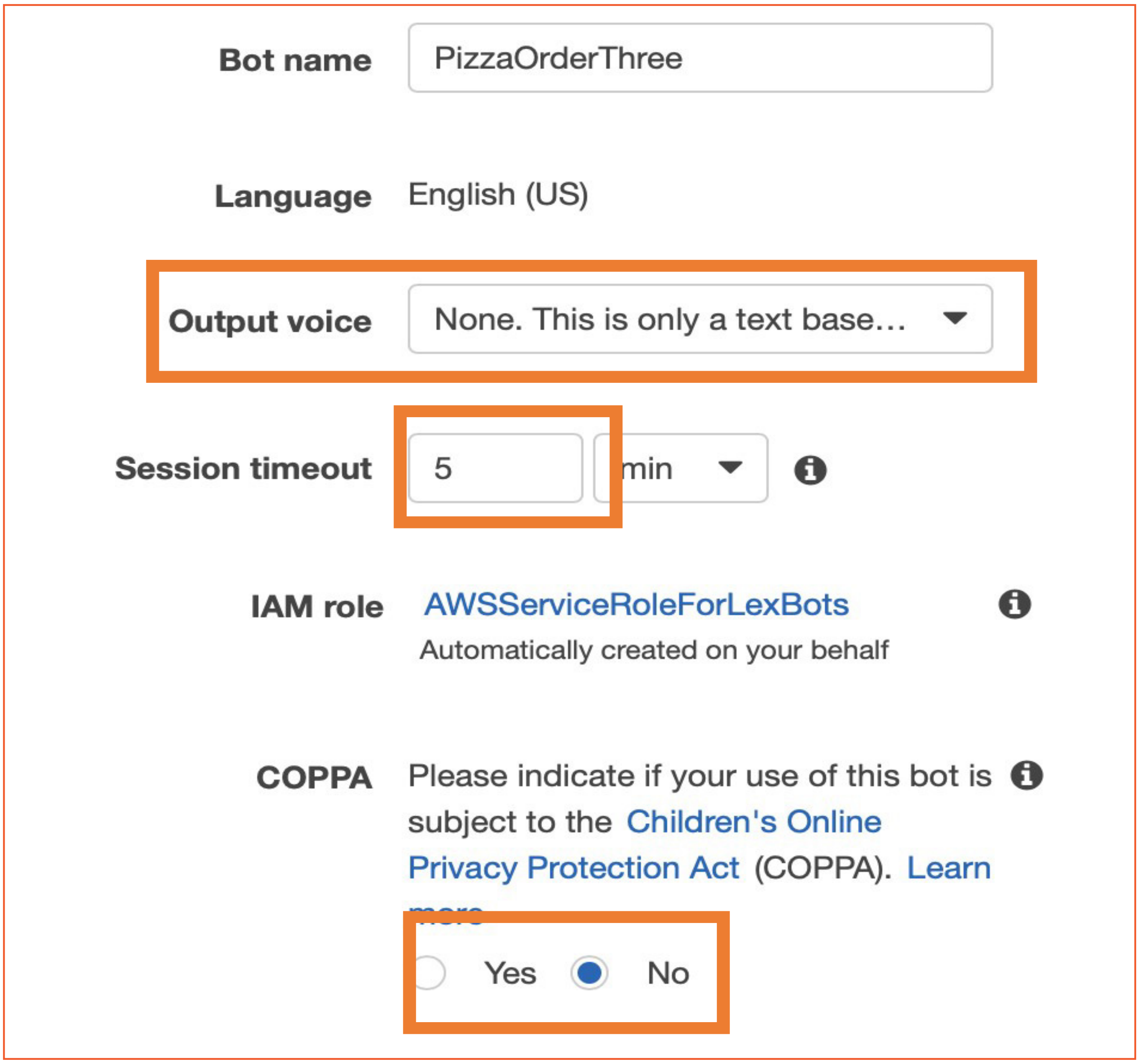
- You can also click Machine Learning and then click Amazon Lex.
- The Amazon Lex product page will now be displayed.
- This page provides more information about Amazon Lex.
- Read the Amazon Lex description above the Get Started button on the bottom of the page as a class.

- Review the key vocabulary:
- Intent: The intent is the goal that the user wants to achieve. "Why did I start this
- chatbot? What do I want the chatbot to be able to do?"
- Utterances: These are phrases that the user types or speaks to the chatbot. For
- example, "Alexa, what's the weather? Alexa, what time is it?"
- Prompts: These are questions the chatbot will ask the user to gather more information. For example, if you ask Alexa what the weather is, she will need to know where you are.
- Slots: These represent data the user provides to the chatbot in response to prompts.
- Confirmations: These confirm the information the user has input; each typically requires a yes/no response from the user.
- Fulfillment: This is the logic needed to complete the intent.
- Click the Get Started button.

Step 5. Set up your chatbot. Click Custom Bot on the left side of the main menu.



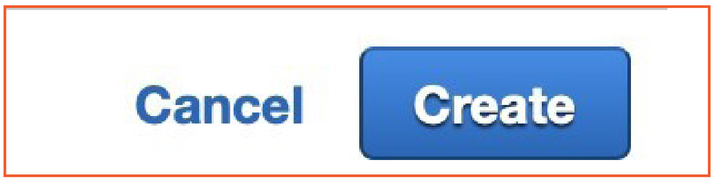
- Name the bot OrderPizza. (Note: There are no spaces in the name.)
- Make the following initial settings:



Bot name	PizzaOrderThree
Language	English (US)
Output voice	None. This is only a text base...
Session timeout	5 min
IAM role	AWSServiceRoleForLexBots Automatically created on your behalf
COPPA	Please indicate if your use of this bot is subject to the Children's Online Privacy Protection Act (COPPA). Learn more Yes [✓ No]

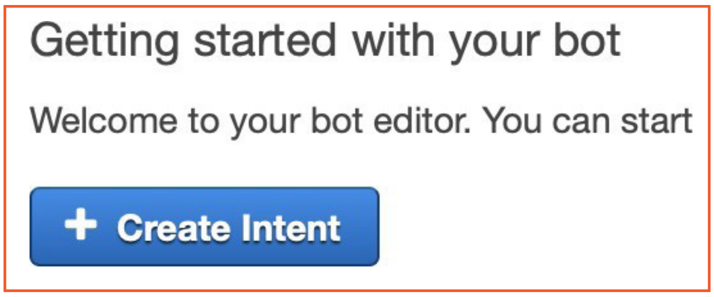
- Select None for voice output. Students can change this later on their own if they so choose.
- Select 5 Minutes for the session timeout. This is how long the bot should wait before ending the chat session.
- Select No under “COPPA: Please indicate if your use of this bot is subject to the Children’s Online Privacy Protection Act (COPPA).” (Note: We are not creating a bot that will gather private information on children younger than 13.)

- Click the blue Create button.



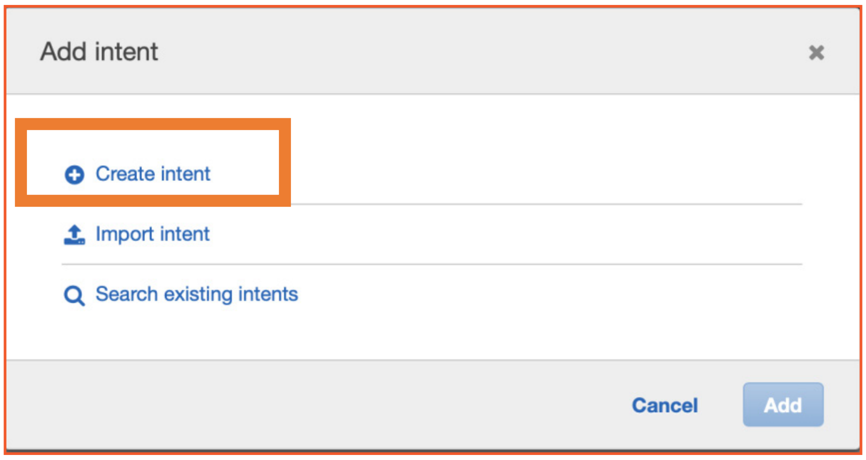
Step 6. Create your first intent.

- Click the blue + Create Intent button.

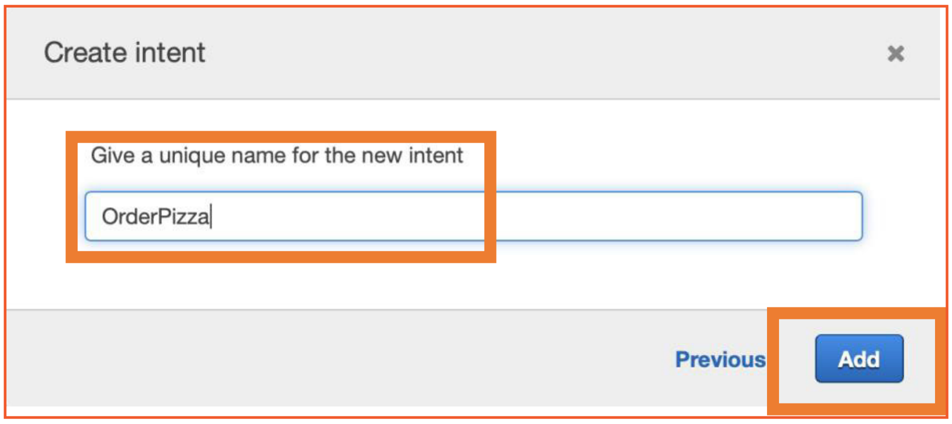

Getting started with your bot

Welcome to your bot editor. You can start

- A new pop-up window will appear.



- Click + Create intent.
- Title the intent OrderPizza (no spaces).
- Click the blue Add button.

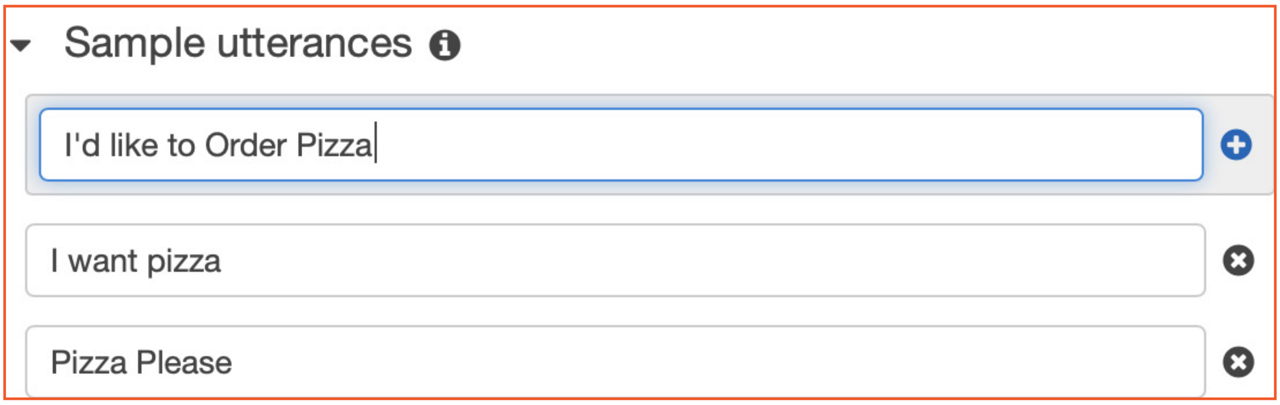

Create intent

Give a unique name for the new intent

OrderPizza

Step 7. Build your utterances. These are phrases the user will use to start the conversation with the chatbot.

- Think about what the intent means. Use those brainstorms as we build utterances, slots, and fulfillments.
- Type the following utterances:
  - o I'd like to Order Pizza
  - o I want pizza
  - o Pizza Please


Sample utterances

I'd like to Order Pizza

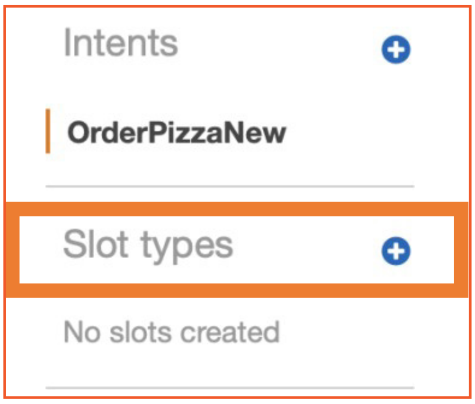
I want pizza

Pizza Please

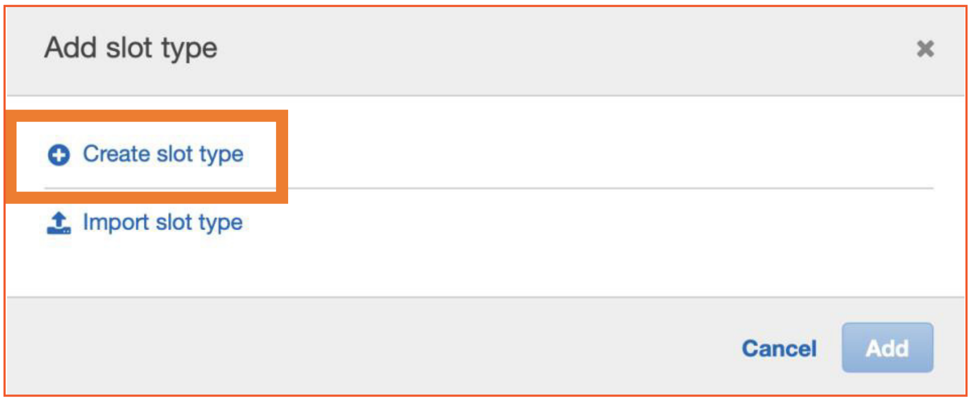
- Create one or two more utterances of your choice.

Step 8. Build your first slot.

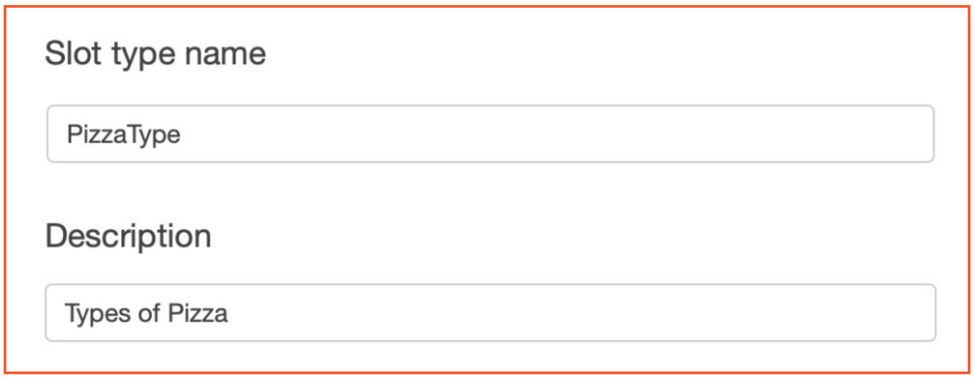
- Look at the left side of the console in the editor.
- Click + next to Slot types.



- Click + Create slot type.



- Title the slot PizzaType (no spaces).
- Describe the slot as Types of pizza.



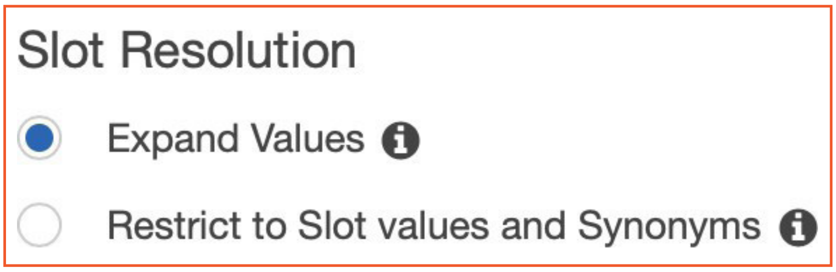
Slot type name

PizzaType

Description

Types of Pizza

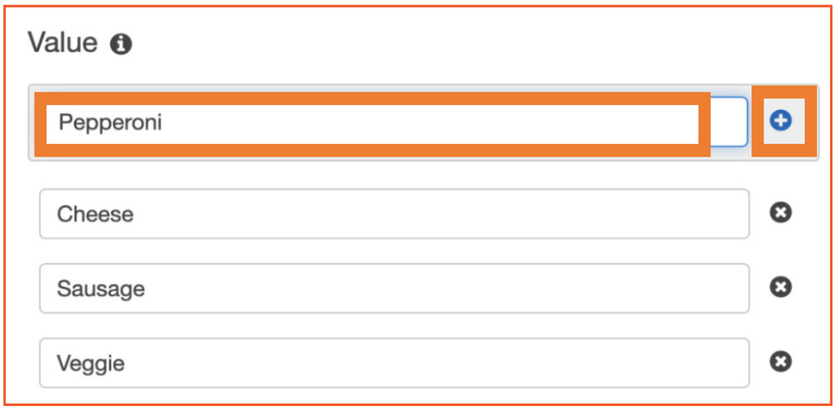
- Keep the default Expand Values selected.


Slot Resolution

[✓ Expand Values]

Restrict to Slot values and Synonyms

- Add additional values in the Value section by hitting the + sign to add additional rows. These are the different types of pizza users can order.


Value

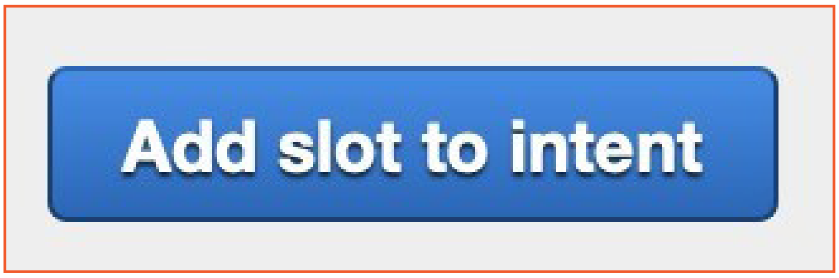
Pepperoni

Cheese

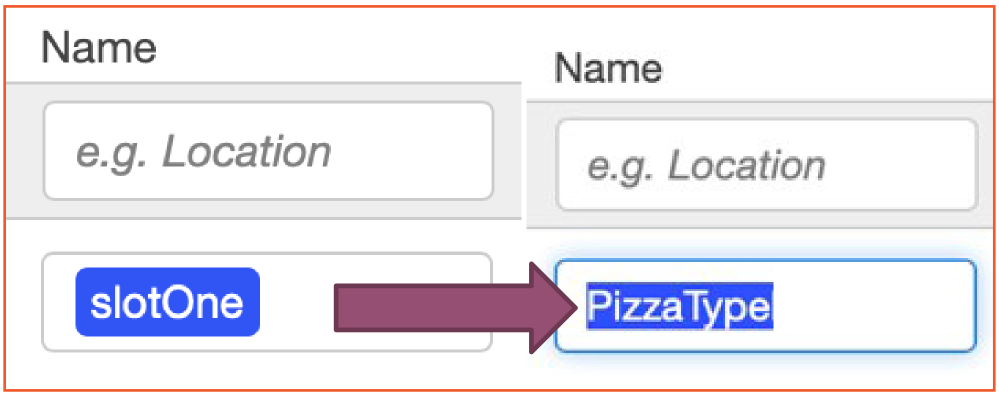
Sausage

Veggie

  - o Cheese
  - o Pepperoni
  - o Veggie
  - o Chicken
  - o One or two more types of pizza
  - o Click Add slot to intent.

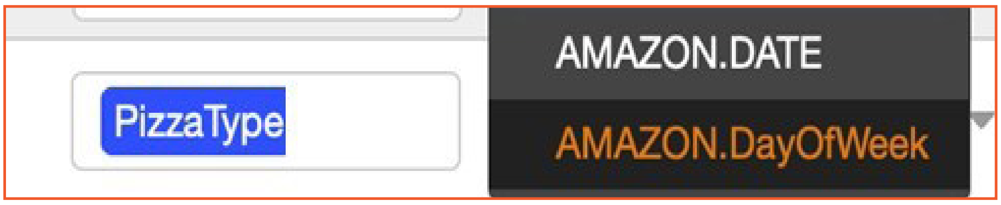


- You should see your slot added to the intent under the slot heading in the dashboard as slotOne.
- Click in the box with slotOne to rename this PizzaType (no spaces).



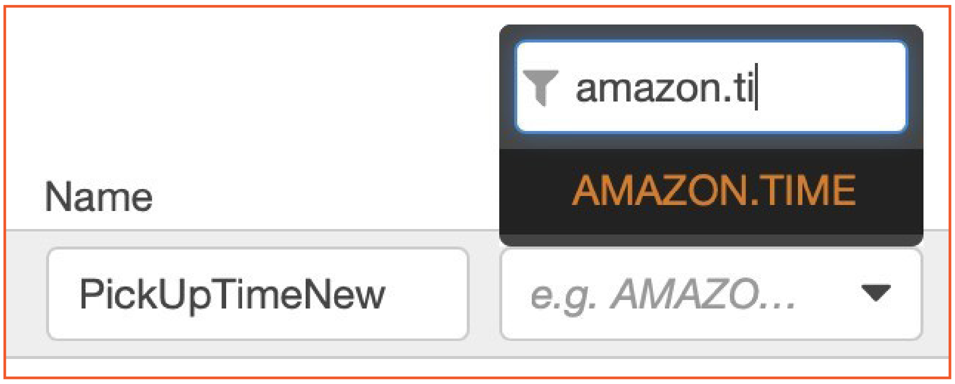
- Change the prompt from "What city?" to "What type of pizza would you like?" This is telling the chatbot to ask the user what type of pizza they would like when they type the original utterance.

Step 9. Build your second slot.

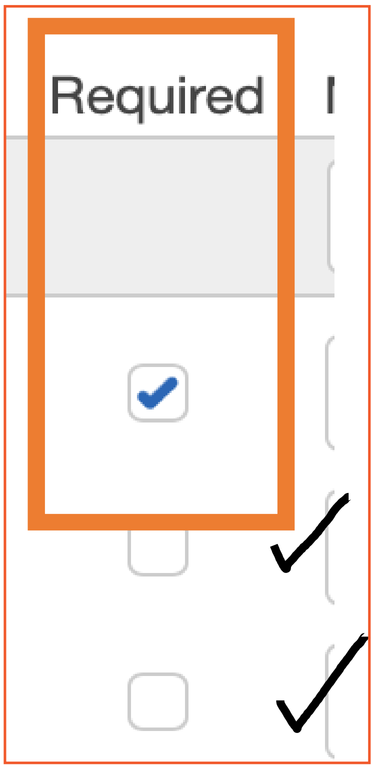


- Scroll down on the screen to see your first slot.
- In the gray box above your first slot, type a new title for your second slot.
- Title the slot PickupDate (no spaces).
- Select the dropdown menu under Slot Type.
- Scroll down to choose AMAZON.DATE from the dropdown menu. You can also search.
- Type the prompt "When would you like your pizza?"
- Click the blue + button at the end of the row to add the slot.

Step 10. Build your third slot.

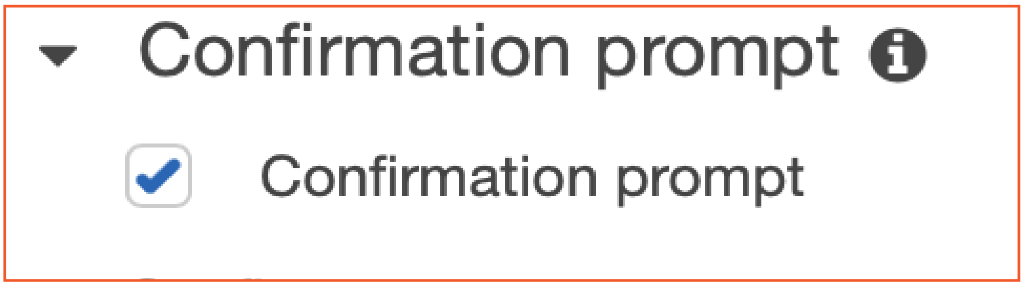


- Scroll down on the screen to see your first slot.
- In the gray box above your first slot, type a new title for your third slot.
- Title the slot PickUpTimeNew (no spaces).
- Choose AMAZON.TIME from the dropdown menu.
- Type the prompt "What time would you like to pick up your pizza?"
- Click the blue + button at the end of the row to add the slot.
- Make all three slots required by clicking the Required checkbox next to each slot.

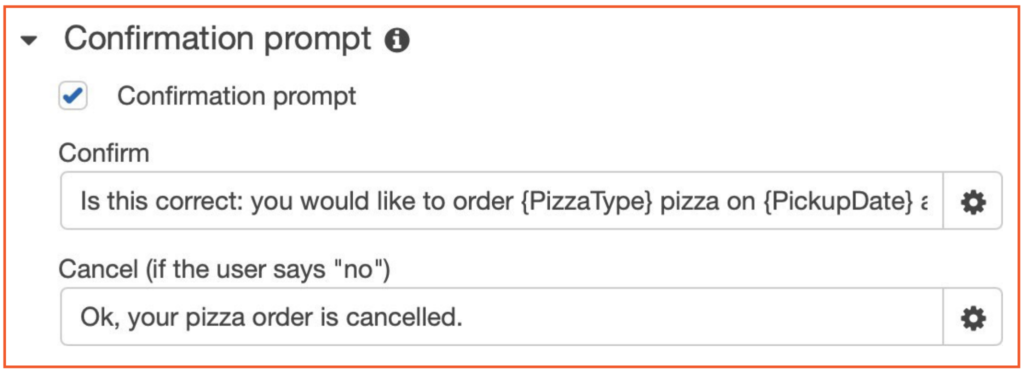


Step 11. Create a confirmation prompt. This lets the user know that the action they have requested has been completed.

- Scroll down under Slots to the Confirmation prompt section.
- Select the checkbox next to Confirmation prompt.



- Use the names of the slots you created: PizzaType, PickupDate, and PickupTime.
- Type "Is this correct: you would like to order {PizzaType} pizza on {PickupDate} at {PickupTime}?"



Confirm

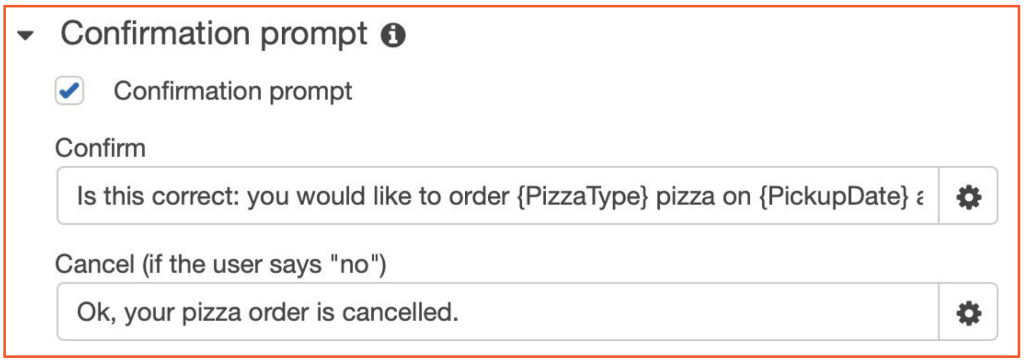
Is this correct: you would like to order {PizzaType} pizza on {PickupDate} a

Cancel (if the user says "no")

Ok, your pizza order is cancelled.

Note: Be sure to type the slot names exactly as they are typed in the slots section. All words, capitalizations, and spacings must match exactly between confirmation and slots.

- Set the "No" answer to a response of Ok, your pizza order is cancelled.



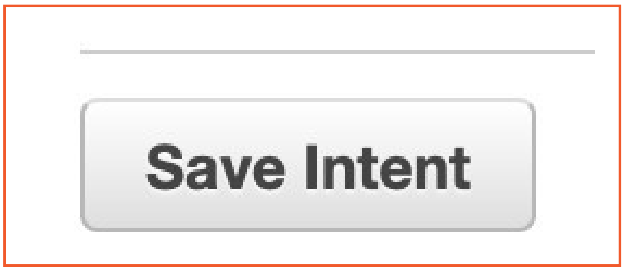
Confirm

Is this correct: you would like to order {PizzaType} pizza on {PickupDate} a

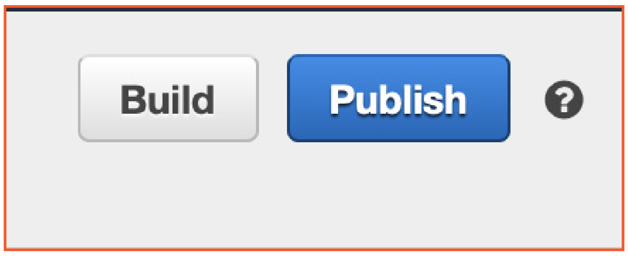
Cancel (if the user says "no")

Ok, your pizza order is cancelled.

- Leave all other settings at their defaults.
- Save your intent by clicking Save Intent at the bottom of the screen.



- Build your bot by clicking Build in the upper-right corner.



- The build process could take a minute.
- Test your bot.
- Click Test Chatbot on the far-right side of the screen—a new side pane will open to test the bot.
- Have a partner test your bot by typing in the new side pane.
- Note:
- Be sure to type out the entire date (for example, May 31, 2019).
- Be sure to type out the time (for example, four p.m. or noon).

Be aware of these troubleshooting tips and tricks:

- Be sure to save each slot and intent immediately, or the data will be lost.
- The confirmation command must have slots typed exactly as they are in slots. Be sure all words are consistent in upper- and lowercase letters and spacings.

1.4 CONVERSATIONAL AI INTERFACE

In this project you will build a conversational AI interface to Covid-19 data using 'SAP Conversational AI'

It aims to familiarize you with building conversational AI bots (a.k.a chatbots) with voice and text interfaces to solve a real-world problem.

THE PROBLEM

Governments and private enterprises generate a lot of data that is publicly accessible. The government publishes data in a number of areas that are of significant public concern such as health indicators, list of hospitals and primary care centers, weather data and warnings, vaccinations, access to public services such as banks and others.

We now take one example of such an area that is of significant impact to the citizens. The central and state governments publish data about COVID-19. This data is available on the Department of Public Health websites. However, certain sections of the society such as citizens with visual or hearing impairments find it difficult to access the data.

The aim of this project is to build a chatbot using a voice interface to make it possible for those with visual impairments to access the data.

GOALS OF THE PROJECT

At the end of the project, the student will be able to

1. Implement voice and text chat applications using popular frameworks
2. Convert text queries to database queries
3. Use sensors such as GPS to augment the app functionality
4. Handle queries that require statistical and ML techniques
5. Solve real-world challenges such as variations in sentence structure and idiomatic usage of language

PRE-REQUISITE

1. Database and SQL concepts as explained in the Database and SQL sections of the basic and step-up modules
   a. How to use SELECT and GROUP BY to query data
   b. How to use mathematical functions such as aggregations, min, max and average

2. Statistics concepts as explained in the Statistics section of the basic and step-up modules
    a. Concepts such as how to calculate min, max, average

3. The Time series forecasting as explained in the forecasting section of the basic and step-up modules
    a. Train and predict with a forecasting model using methods such as exponential smoothing

4. Machine learning models for Text Processing as explained in the text processing section of the basic and step-up modules
    a. How to train and use an OCR model or service
    b. How to use text summarization

5. Have a good understanding of at least one popular conversational AI framework. The following links provide tutorials for popular frameworks
    a. SAP Conversational AI - https://cai.tools.sap/, can be deployed to Google Home, Alexa and Cortona.
    b. Alexa - https://developer.amazon.com/en-IN/alexa/alexa-skills-kit, https://developer.amazon.com/en-GB/alexa/alexa-voice-service/learn/tutorials
    c. Google Home - https://developers.google.com/assistant/smarthome/develop/create
    d. Cortona - https://developer.microsoft.com/en-us/cortana/

TUTORIALS TO LEARN ABOUT BUILDING CHATBOTS

EXPLORE

https://developers.sap.com/tutorials/cai-bot-getting-started.html

https://cai.tools.sap/blog/build-your-first-faq-chatbot-with-sap-conversational-ai/

https://chatbotsmagazine.com/quick-start-develop-a-chat-bot-with-aws-lex-lambda-part-1-b6f7c80ebba6

EXPLORE

https://hackernoon.com/building-a-serverless-chatbot-with-aws-lex-lambda-and-amazon-aurora-part-1-5406e8db6123

https://cloud.google.com/community/tutorials/ios-chatbot

DOWNLOAD THE DATA

1. All India COVID data is available from the Indian Statistical Institute, Bangalore website -https://www.isibang.ac.in/~athreya/incovid19/data.html
2. Government bulletins and travel advisories can be downloaded from the Ministry of Health and Family Welfare websites.

THE SOLUTION

How to proceed towards a solution:

1. Create an account in one of the popular frameworks listed above
2. The solution in general consists of a backend database containing the COVID-19 data, a chatbot service to receive requests and respond with answers and a client to get user input and show the answers
3. Download the data and store it in a database
4. Train the chatbot to recognize a set of pre-defined questions
5. In case of OCR and text summarization, services from popular cloud providers can be used.
6. Deploy the chatbot

Some sample questions:

1. What is the number of active cases in my current location?
2. What is the number of active cases in Delhi?
3. What is the current case load in Delhi?
4. What is the average number of new cases in the last 10 weeks in Karnataka?

5. What is the rate of increase of confirmed cases in Mumbai?
6. What is the forecasted rate of increase for Delhi for the next 3 weeks?
7. Think of variations of the same question, idiomatic usage, etc

For example, the first question in the previous slide can be rephrased as “What is the weekly average of new cases in Karnataka?”. The app should be able to respond to both variations with the same answer.

Text summarization

1. Every state has issued SOP for travel
2. People find it very difficult to identify the SOP for their particular travel, for example, “What is the quarantine requirement if I am 40 years of age and travelling by flight from Delhi to Bangalore?”
3. Travel advisories and SOPs are published by each state as PDFs on the respective health department websites
4. Use OCR and text summarization to summarize these PDFs
5. Extend the app to provide answer to questions such as “What is the latest travel advisory for Karnataka?”. In this case, provide the summary of the latest travel advisory from Karnataka government.

Solution Outline

This solution outline provides details about one possible solution to the problem using SAP Conversational AI. Students are encouraged to attempt to solve the problem without looking at the solution outline. However, if they get stuck, the outline can provide valuable hints to help them proceed.

1. Create an account – Go to https://cai.tools.sap/ and Sign Up.
2. Familiarize yourself with the steps for building a bot - https://cai.tools.sap/blog/build-your-first-bot-with-sap-conversational-ai/
3. Create an SAP Cloud Platform (SCP) Trial Account
4. Create a Postgres database instance
5. Load the data from Indian Statistical Institute into the Postgres database
6. Follow the steps in https://cai.tools.sap/blog/nodejs-chatbot-movie-bot/ with the following changes
   a. Connect to the Postgres database on SCP instead of the movie database
   b. Connect to the OCR and Text services on SCP

TOPIC 2 - ADVANCED PROJECTS FOR AI

2.1 IMAGE PROCESSING PROJECT

This project is about Identifying most vulnerable fresh water bodies using ML Image Processing techniques.

It aims to make you familiar with training and using image processing models for a real-world problem.

THE PROBLEM

A major problem in India is access to fresh drinking water. Fresh water sources such as rivers and lakes are drying up at an unprecedented pace due to various reasons including climate change, pollution and overuse. Policy makers need data-driven evidence to formulate policies to protect freshwater resources.

Satellite images are available that show the lakes across India. These satellite images are available for a number of years. The task is to identify the increase or decrease in the area of these lakes across time. Lakes that have the highest decrease in area across time can be identified. These lakes can be selected for conservation efforts.

GOALS OF THE PROJECT

At the end of the project, the student will be able to

1. Prepare real-world training and test image datasets
2. Train a deep learning model for image segmentation
3. Apply the model to get predictions
4. Tackle real-world challenges in preparing image datasets, such as, ensuring correct image boundaries and overlapping tiles.

PREREQUISITES

The Student should have

1. Good understanding of image classification and object detection

2. A good understanding of tensorflow

EXPLORE

- Try out the tutorials at

https://www.tensorflow.org/tutorials/images/classification

https://www.tensorflow.org/tutorials/images/segmentation

to learn or brush up the knowledge on Tensorflow.

TUTORIALS TO LEARN ABOUT OBJECT DETECTION

The following tutorials and articles can be used by the student to understand how to train and apply object detection models.

https://www.datacamp.com/community/tutorials/object-detection-guide

https://towardsdatascience.com/airplanes-detection-for-satellite-using-faster-rcnn-d307d58353f1

https://medium.com/intel-software-innovators/ship-detection-in-satellite-images-from-scratch-849ccfcc3072

https://www.pyimagesearch.com/2018/05/14/a-gentle-guide-to-deep-learning-object-detection/

https://towardsdatascience.com/data-science-and-satellite-imagery-985229e1cd2f?gi=e93ba19f0a56

https://medium.com/data-from-the-trenches/object-detection-with-deep-learning-on-aerial-imagery-2465078db8a9

DOWNLOADING THE SATELLITE IMAGES

EXPLORE

1. Open the link in a web browser

https://bhuvan-app3.nrsc.gov.in/data/download/index.php

2. Register as a new user if this is your first visit to the site
3. Select category “Theme/Products”
4. Select theme “Land and Terrain”
5. Select product “OCM: Surface Water Layer Products_2D”
6. Select year 2015 and month Jan
7. Download the image for period “Jan01to02-2015”
8. Select year 2016 and month Jan
9. Download the image for period “Jan01to02-2016”
10. Unzip the image ZIP files

Each satellite image covers the entire Indian sub-continent. The images are in black and white. Water bodies such as lakes, rivers and reservoirs are shown in white while the rest of the surface is black.

THE SOLUTION

How to proceed towards the solution:

1. Use the Jan 2015 image as training and test data.
2. Create a tensorflow object detection model to detect lakes and their boundaries
3. Use the trained model to predict lake boundaries in the Jan 2016 image
4. The lake boundaries may be simple bounding boxes and may go as complex as image masks. Bounding boxes are the simplest to train and predict but they are approximations of the lake boundaries. On the other hand, image masks are the most accurate since they identify the boundaries at individual pixel level. However, they are very complex to annotate, train and predict.
5. Compare the area covered by the boundaries for the same lake between Jan 2015 and Jan 2016. You can calculate the areas for a number of lakes and determine the lakes with the highest reduction in area. These are the lakes that are most vulnerable.

This solution outline provides details about one possible solution to the problem. Students are encouraged to attempt to solve the problem without looking at the solution outline. However, if they get stuck, the outline can provide valuable hints to help them proceed.

1. Install the required software and libraries

   a. Install Python 3
   b. Setup tensorflow for object detection. Follow the steps in https://christopherstoll.org/2018/12/12/tensorflow-object-detection-mac-setup.html
   c. Install OpenCV library

2. Create the training and test images
   a. Use OpenCV to split the large Jan 2015 image into smaller images. For example, you can create 100x100 pixel tiles. Ensure that you use PNG as the output format.
   b. Each tile will contain 0, 1, or more water bodies.
   c. Choose at least 100 images for training and 20 images for testing. The time taken to annotate the images will more if you choose more images. However, you need a sufficiently large number of images to properly train the model.

3. Use LabelImg tool to create bounding boxes on lakes in the images.
   a. Save the output in Pascal VOC format
   b. You will get 1 XML file per image.
   c. The XML file will contain the pixel coordinates of the bounding boxes

4. Generate TFRecord files.
   a. You can follow the steps in https://github.com/tensorflow/models/blob/master/research/object_detection/create_pascal_tf_record.py

5. The images all belong to just one class. Create a label map with just this one class.

6. We will use transfer learning to speed up the training process. Use An object detection training pipeline from https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/configuring_jobs.md. You can also find sample config files at https://github.com/tensorflow/models/tree/master/research/object_detection/samples/configs. "ssd_mobilenet_v2_coco.config" is a good starting point.

7. Run the training locally (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/running_locally.md)

8. It is also recommended during the training to start the evaluation job. You can then monitor the process of the training and evaluation jobs by running Tensorboard on your local machine.

9. After finishing with training, export the trained model to a single file (Tensorflow graph proto)

10. You can now use the model for inference

11. Create the images for inference by selecting the Jan 2016 image and splitting it into tiles just like you did for the Jan 2015 image.

12. Ensure that you split the image such that each tile from Jan 2016 covers the exact same portion of the map as the corresponding tile from Jan 2015.

13. Run the inference using the trained model

14. You will now get the bounding boxes for the detected lakes. You can calculate the areas and compare them to the corresponding areas calculated from the 2015 image.

2.2 ML USING DATA SCIENCE

This project involves a complex balance between COVID-19 spread and urbanTransportation. In this project you will attempt an impossibly difficult task of balancing mobility and infection spread, that if achieved, will benefit the entire community.

THE PROBLEM

The COVID-19 pandemic crisis has forced public transport to completely shut down. And has prevented people from moving around for their livelihood. In Bengaluru, around 41% population uses the public transport buses to move around

The public transport during an epidemic or a pandemic is shut down partially or completely to prevent spread of the infection. And this is important in densely populated cities where public transport can quickly spread the infection

So, is it possible to operate the buses in a safe manner, without spreading the infection? And, this is the question we will try to answer using data science and epidemic models

Our goal is to leverage data science to enable effective operation of Bengaluru Metropolitan Transport Corporation (BMTC) as well as contain spread of infection in Bengaluru City

GOALS OF THE PROJECT

We will achieve our objective by

1. Identifying the programming language for the project
2. Setting up the tools necessary to build, test and deploy
3. Identify and understand data requirements
4. Collecting the data
5. Understanding Origin-Destination (OD) flow matrix for Urban Mobility
6. Preparing the data
7. Identifying the epidemic model
8. Processing the data and testing the model
9. Running the simulations
10. Publishing the project on Heroku for access by anyone on the Internet

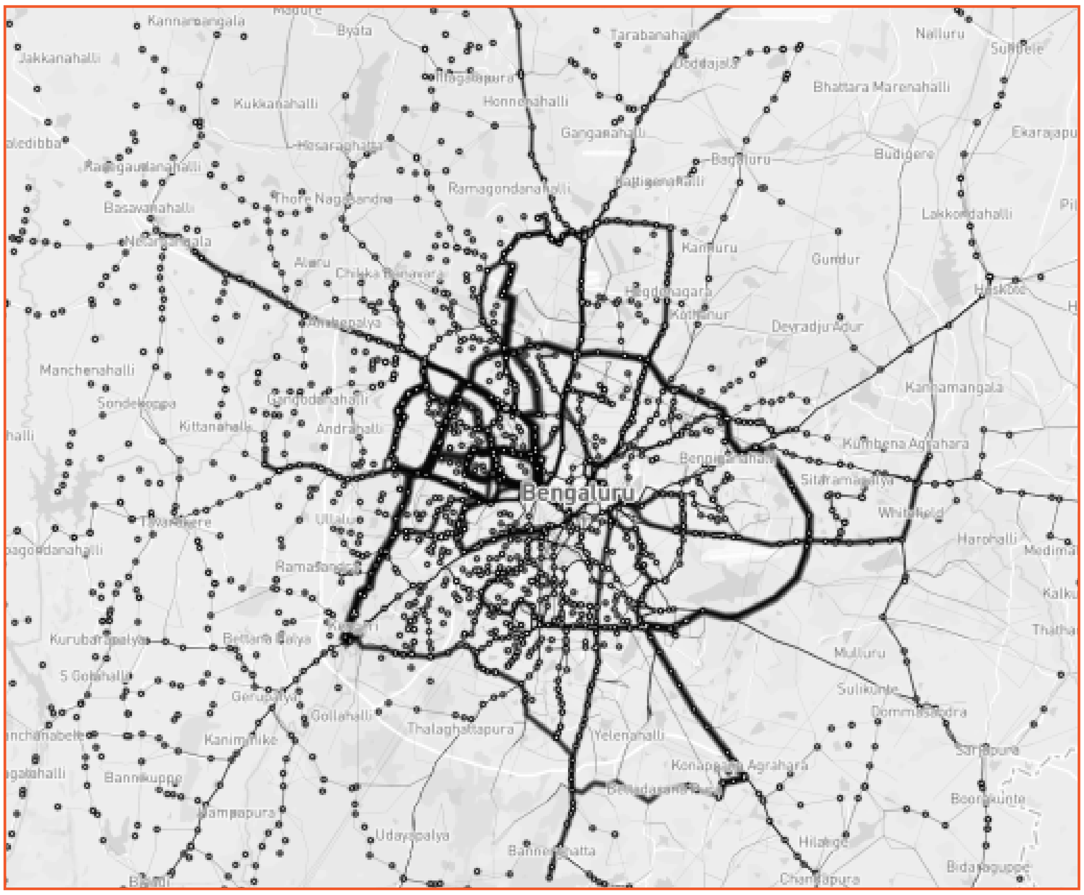
PRE-REQUISITES

Data Requirements

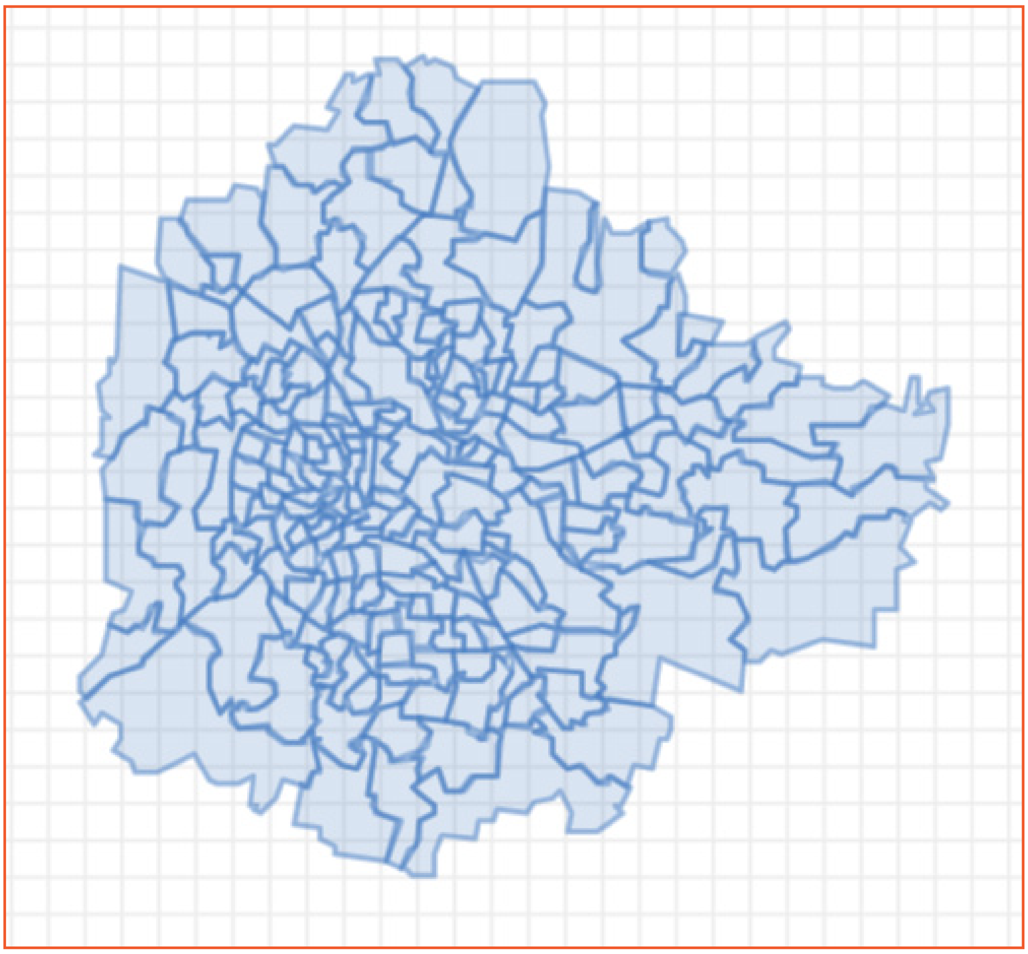
Now, that we have the tools setup and running, lets identify the data required for the project. Our objective is to operate buses safely in Bengaluru. So, we need data about all buses operated by Bengaluru's city transportation operator, BMTC. And we also need data about Bengaluru. Note that Bengaluru city is managed by BBMP, a city corporation and the city is divided into WARDs. Each 'ward' has a clearly defined geographic boundary, and has data about its population. So, let's get the following data:

- BMTC data (Routes, Bus Stops, ...)
- BBMP ward details (Zone, Number, Population, Geo Boundaries)
- Ward wise infection data

BMTC data can be found at https://github.com/geohacker/bmtc. Data is available for 2045 routes with information about bus stops, the latitude and longitude of each bus stop, the sequence of stops, etc



BBMP data can be found at https://github.com/openbangalore/bangalore. And, the ward wise population data from https://indikosh.com/city/708740/bruhat-bengaluru-mahanagara-palike



Note, that the BBMP ward and their GEO boundaries are required, so that we can locate all the bus stops within each ward. This is required to create the Origin-Destination flow matrix, for urban mobility which I will explain in a bit

Infection data is not readily available. And the one I could find was at https://indianexpress.com/article/cities/bangalore/covid-19-101-cases-in-bengaluru-so-far-heres-the-list-of-wards-affected-6368503/

The project will use a Python language and libraries.

THE SOLUTION

Origin-Destination (OD) flow matrix for Urban Mobility

So, to analyze the spread of infection, we need data about the number of people moving between these wards in a given day, using buses. So, how can we get this information?

Well, we have latitude and longitude of every bus stop, so we can find out all the bus stops in a WARD, and as we also have the bus stop sequence of the bus route, we know which is the previous and the next bus stop. And using this information, we can find all the origin and destination bus stops. Let's start creating the OD flow matrix. All the code shown below is available on GitHub

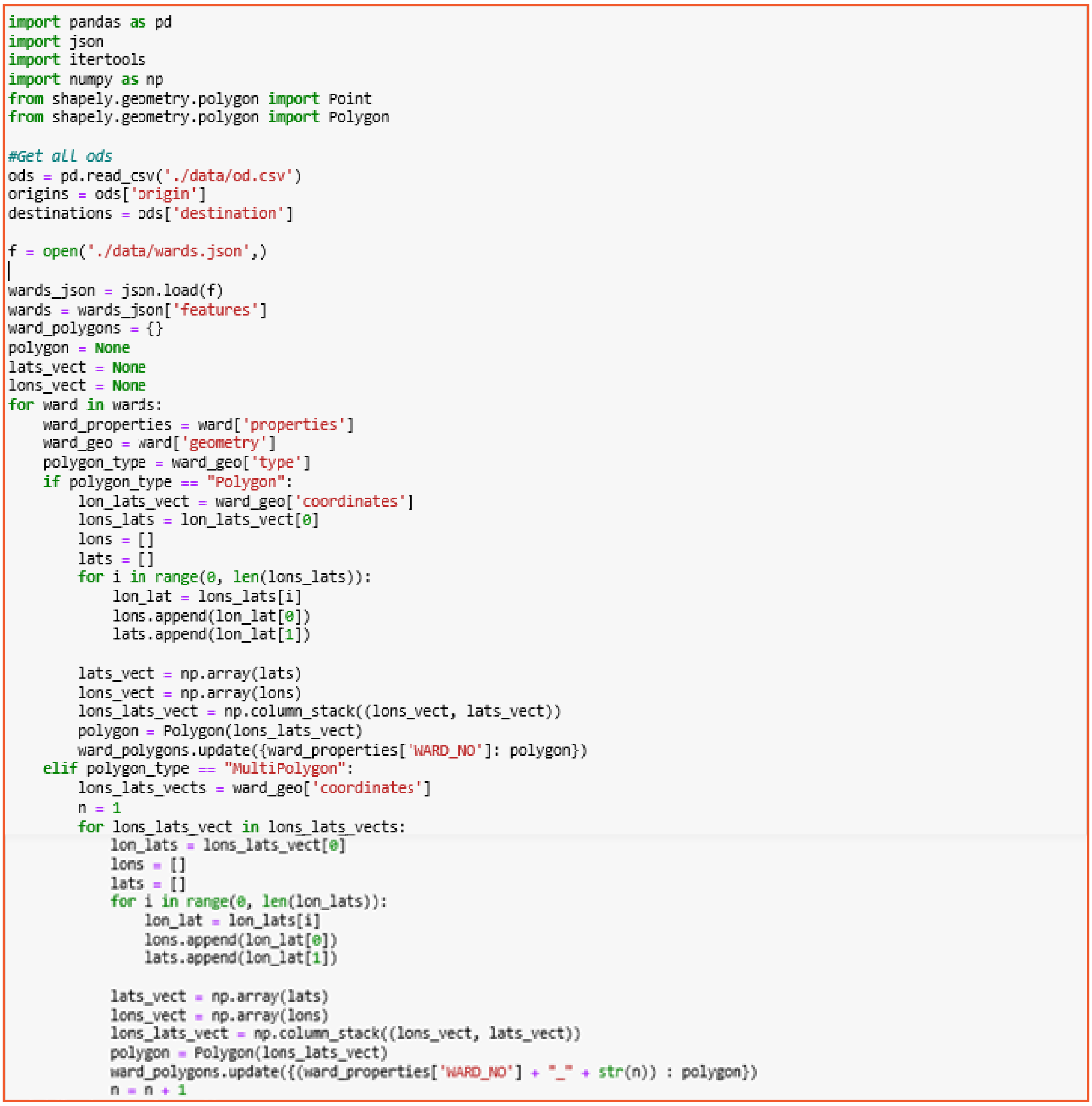
https://github.com/arcexe/btcat

Notebook – 1 "covid19_data_preparation"

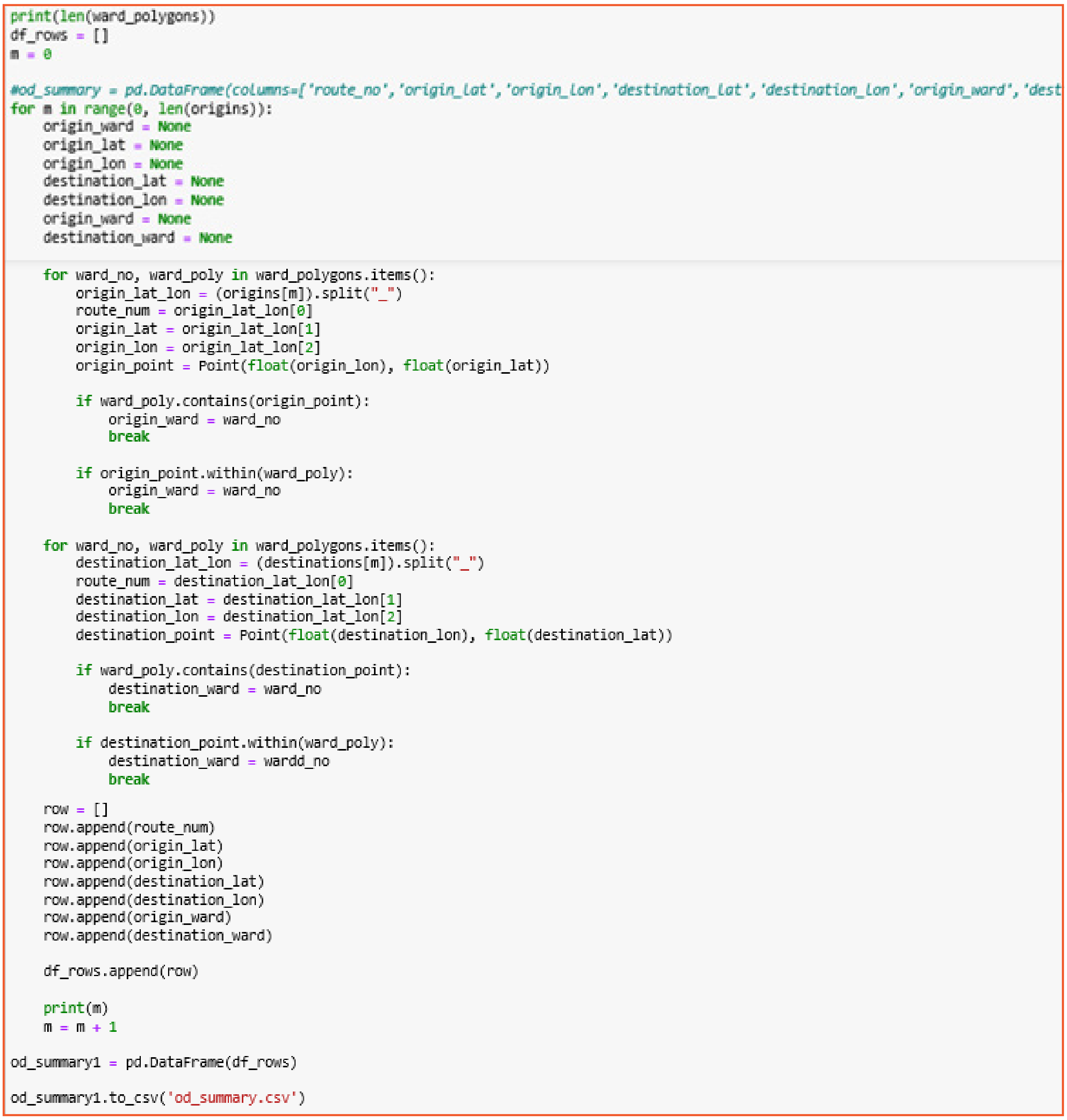
```python
import pandas as pd
import json
import itertools

#Get all bus routes and organize them in arrays
bus_routes = pd.read_csv('./data/bmtc_dump.csv')
route_nums = bus_routes['route_no']
all_bus_route_stops = bus_routes['map_json_content']
#print(all_bus_route_stops)
all_routes = []

for i in range(0, len(all_bus_route_stops)):
    routes_json_array = json.loads(all_bus_route_stops[i])
    route_num = route_nums[i]
    routes = []
    for item in routes_json_array:
        ll_array = item['latlons']
        x = route_num + "_" + ll_array[0] + "_" + ll_array[1]
        routes.append(x)

    all_routes.append(routes)

print(len(all_routes))
od_combinations = []
for route in all_routes:
    for subset in itertools.combinations(route, 2):
        od_combinations.append(subset)

df = pd.DataFrame(od_combinations)

#saving the dataframe
df.to_csv('od.csv')

print(df)
```

In the code above, we read the BMTC data which is a CSV file using the Pandas read_ csv () function. And then we get all the route numbers and their bus stops. And in the for loop we iterate through each bus stop of the route and get the latitude and longitude. Now we create origin and destination combination between all possible bus stops using the itertools. combinations () function

Finally, the OD combinations are stored to od.csv file



```python
import pandas as pd
import json
import itertools
import numpy as np
from shapely.geometry.polygon import Point
from shapely.geometry.polygon import Polygon

#Get all ods
ods = pd.read_csv('./data/od.csv')
origins = ods['origin']
destinations = ods['destination']

f = open('./data/wards.json',)

wards_json = json.load(f)
wards = wards_json['features']
ward_polygons = {}
polygon = None
lats_vect = None
lons_vect = None
for ward in wards:
    ward_properties = ward['properties']
    ward_geo = ward['geometry']
    polygon_type = ward_geo['type']
    if polygon_type == "Polygon":
        lon_lats_vect = ward_geo['coordinates']
        lons_lats = lon_lats_vect[0]
        lons = []
        lats = []
        for i in range(0, len(lons_lats)):
            lon_lat = lons_lats[i]
            lons.append(lon_lat[0])
            lats.append(lon_lat[1])

        lats_vect = np.array(lats)
        lons_vect = np.array(lons)
        lons_lats_vect = np.column_stack((lons_vect, lats_vect))
        polygon = Polygon(lons_lats_vect)
        ward_polygons.update({ward_properties['WARD_NO']: polygon})
    elif polygon_type == "MultiPolygon":
        lons_lats_vects = ward_geo['coordinates']
        n = 1
        for lons_lats_vect in lons_lats_vects:
            lon_lats = lons_lats_vect[0]
            lons = []
            lats = []
            for i in range(0, len(lon_lats)):
                lon_lat = lon_lats[i]
                lons.append(lon_lat[0])
                lats.append(lon_lat[1])

            lats_vect = np.array(lats)
            lons_vect = np.array(lons)
            lons_lats_vect = np.column_stack((lons_vect, lats_vect))
            polygon = Polygon(lons_lats_vect)
            ward_polygons.update({(ward_properties['WARD_NO'] + "_" + str(n)) : polygon})
            n = n + 1
```



```python
print(len(ward_polygons))
df_rows = []
m = 0

#od_summary = pd.DataFrame(columns=['route_no','origin_lat','origin_lon','destination_lat','destination_lon','origin_ward','dest
for m in range(0, len(origins)):
    origin_ward = None
    origin_lat = None
    origin_lon = None
    destination_lat = None
    destination_lon = None
    origin_ward = None
    destination_ward = None

    for ward_no, ward_poly in ward_polygons.items():
        origin_lat_lon = (origins[m]).split("_")
        route_num = origin_lat_lon[0]
        origin_lat = origin_lat_lon[1]
        origin_lon = origin_lat_lon[2]
        origin_point = Point(float(origin_lon), float(origin_lat))

        if ward_poly.contains(origin_point):
            origin_ward = ward_no
            break

        if origin_point.within(ward_poly):
            origin_ward = ward_no
            break

    for ward_no, ward_poly in ward_polygons.items():
        destination_lat_lon = (destinations[m]).split("_")
        route_num = destination_lat_lon[0]
        destination_lat = destination_lat_lon[1]
        destination_lon = destination_lat_lon[2]
        destination_point = Point(float(destination_lon), float(destination_lat))

        if ward_poly.contains(destination_point):
            destination_ward = ward_no
            break

        if destination_point.within(ward_poly):
            destination_ward = wardd_no
            break

    row = []
    row.append(route_num)
    row.append(origin_lat)
    row.append(origin_lon)
    row.append(destination_lat)
    row.append(destination_lon)
    row.append(origin_ward)
    row.append(destination_ward)

    df_rows.append(row)

    print(m)
    m = m + 1

od_summary1 = pd.DataFrame(df_rows)

od_summary1.to_csv('od_summary.csv')
```

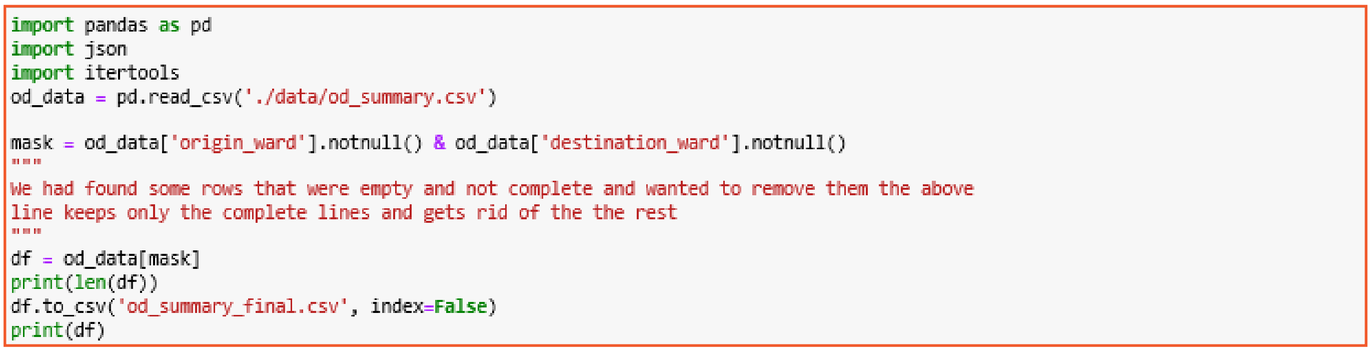
Now that we have generated the routes for Bangalore, we will need to locate every OD combination for every single ward in Bangalore. You can notice on the heatmap on my dashboard about the OD combination which I have called route counts which are the total number OD combinations. This cell can be used to do that.

tip – If you wish to perform and run the cell, keep a note that this would run for around half a day so make sure that you have not made any errors while writing the code

This concludes the first of four notebooks we would be viewing and understanding

Notebook – 2 “covid19_data_processing”

This notebook is the continuation of the notebook “covid19_data_preparation”



```
import pandas as pd
import json
import itertools
od_data = pd.read_csv('./data/od_summary.csv')

mask = od_data['origin_ward'].notnull() & od_data['destination_ward'].notnull()
"""
We had found some rows that were empty and not complete and wanted to remove them the above
line keeps only the complete lines and gets rid of the the rest
"""
df = od_data[mask]
print(len(df))
df.to_csv('od_summary_final.csv', index=False)
print(df)
```

Now that we’ve generated the file od_summary which was made in the last notebook, let’s make use of it

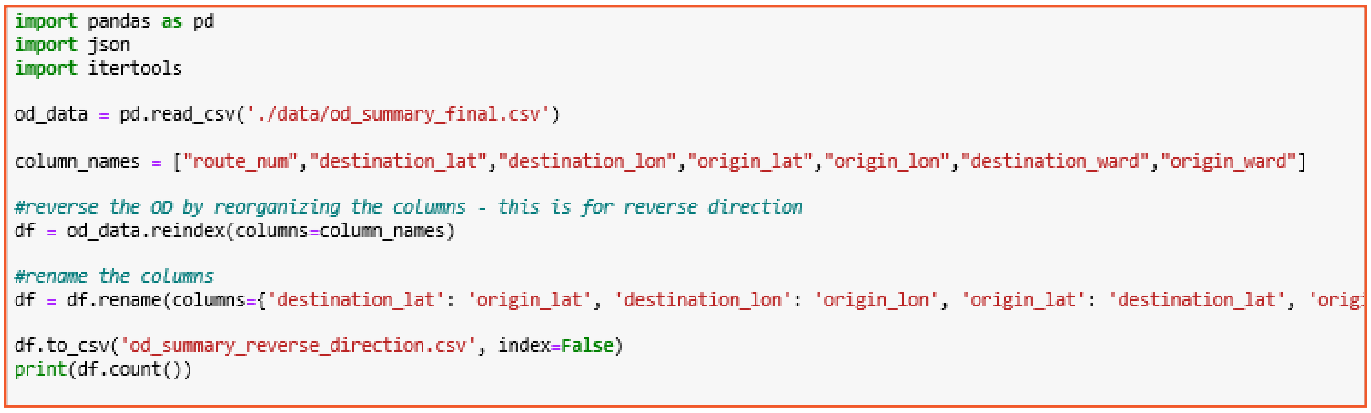
Sometimes data can’t be perfect and have everything present in it, and it was the same case for me as well and in my case we had some rows which were empty and some parts of the columns missing like some rows of the columns; ‘origin_ward’, ‘destination_ward’.

The “notnull ()” function, you guessed it, gets only the lines which are not empty or null.

The above cell filters all the rows which have everything present and nothing missing and stores them as the file “od_summary_final.csv”

csv stands for ‘comma separated values’ which is a file extension. You may have seen some extensions while you were saving a file for example: ‘.txt’, ‘.pptx’, ‘.pdf’.

The “df.to... index=False line gets” rid of the index whilst we are writing the csv. Let’s move on to the next cell



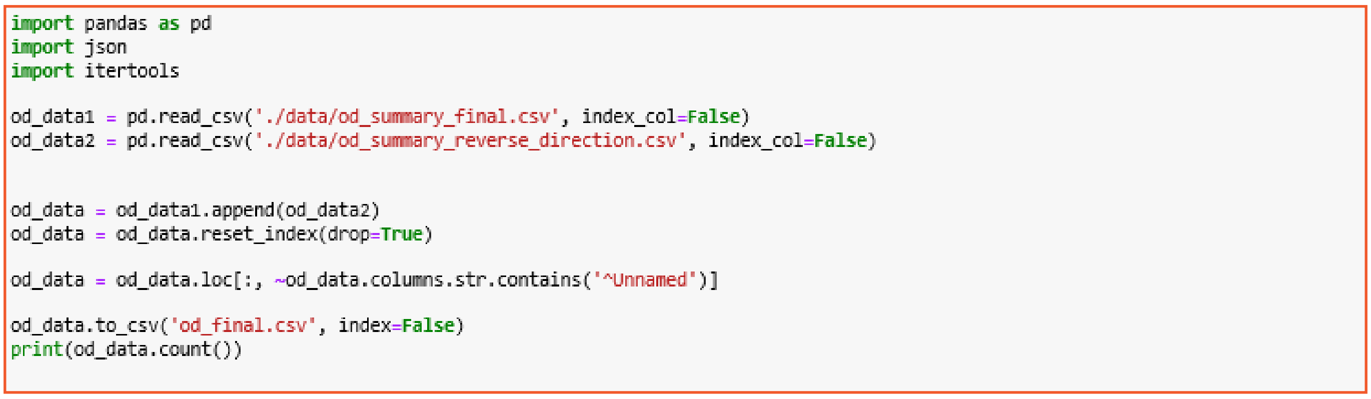
```python
import pandas as pd
import json
import itertools

od_data = pd.read_csv('./data/od_summary_final.csv')

column_names = ["route_num","destination_lat","destination_lon","origin_lat","origin_lon","destination_ward","origin_ward"]

#reverse the OD by reorganizing the columns - this is for reverse direction
df = od_data.reindex(columns=column_names)

#rename the columns
df = df.rename(columns={'destination_lat': 'origin_lat', 'destination_lon': 'origin_lon', 'origin_lat': 'destination_lat', 'orig

df.to_csv('od_summary_reverse_direction.csv', index=False)
print(df.count())
```

(If you cannot view the cell properly, please go to the link mentioned above to see the git hub and view it there)

We are now going to get the file that we had generated in the above cell (using the df.to_csv...) function and load the columns and reindex them, meaning the data gets reversed, what that means is that, let's say for example the origin_ward and destination column and use the reindex function, and now that we have reindexed them the data under origin_ward goes to destination_ward and vice versa and finally we store the data we just got a csv as well

I have used the "rename" function since we have reversed the data and since we do not want wrong column names, this function is used if you were thinking about why I had used this function

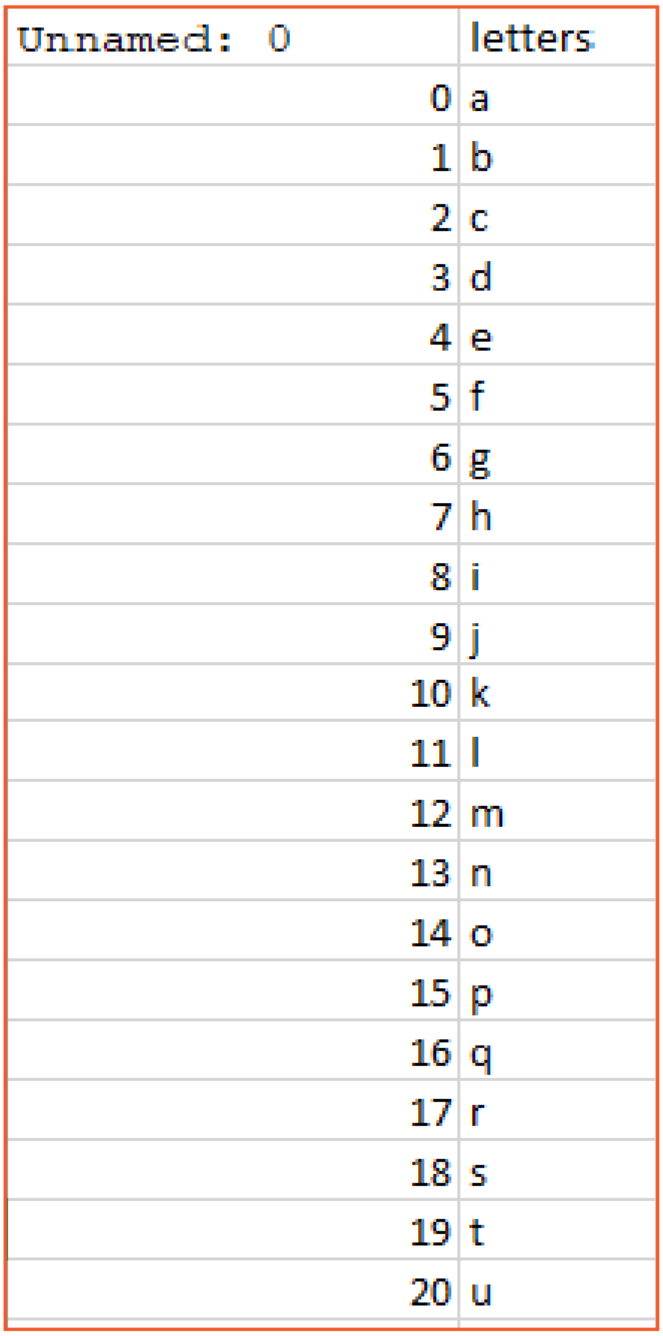
Now you might be wondering why I have made this cell so here's the reason, Since vehicles travel the both ways, we can't assume that people will travel in a single direction so that is why we have performed this action and yes I have stored this as a csv as well since we will be using this later in the below cells. Let's continue



```python
import pandas as pd
import json
import itertools

od_data1 = pd.read_csv('./data/od_summary_final.csv', index_col=False)
od_data2 = pd.read_csv('./data/od_summary_reverse_direction.csv', index_col=False)


od_data = od_data1.append(od_data2)
od_data = od_data.reset_index(drop=True)

od_data = od_data.loc[:, ~od_data.columns.str.contains('^Unnamed')]

od_data.to_csv('od_final.csv', index=False)
print(od_data.count())
```

Now that we have made the reversed direction, let's make a small change which would be helpful in the future. Another note we should make is that whenever we generate csv(s) there is an additional column called '^Unnamed' and since we do not require that I am going to remove it, Let's see how we can do that in the above cell and since we do not want the change we made to be lost we have saved it as csv

This is an example about what is might look like if we open it;



Unnamed: 0	letters
0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p
16	q
17	r
18	s
19	t
20	u

Str.Contains() function is used to test if a pattern is contained within a string of a series or index. It is an index in our case. We have used contains since even we have a pattern of the unnamed column which is that 1 is getting added after every row. Now let's run this and voila we have the additional column removed. Let's move on

```
import pandas as pd
import json
import itertools

od_data = pd.read_csv('./data/od_final.csv')

df = od_data.groupby(['destination_ward','origin_ward', 'route_num'])['route_num'].count().reset_index(name='route_counts')

df = df.rename(columns={'destination_ward': 'ward_no'})

print(df)

df.to_csv('od_final_grouped.csv')
```

"What does this cell do?" you ask. The answer is that it makes use of the groupby () and A groupby operation involves some combination of splitting the object, applying a function, and combining the results.

And that is what we did, first we got some columns and have performed an operation which is that the count or number of 'route_numbers' are equal to the route_counts which I have talked about earlier and that data would be necessary so we have used the reset_index() function in a groupbywhich then results in the operation we performed into a whole new column, we have also renamed the column destination_ward to ward_no.

Next cell

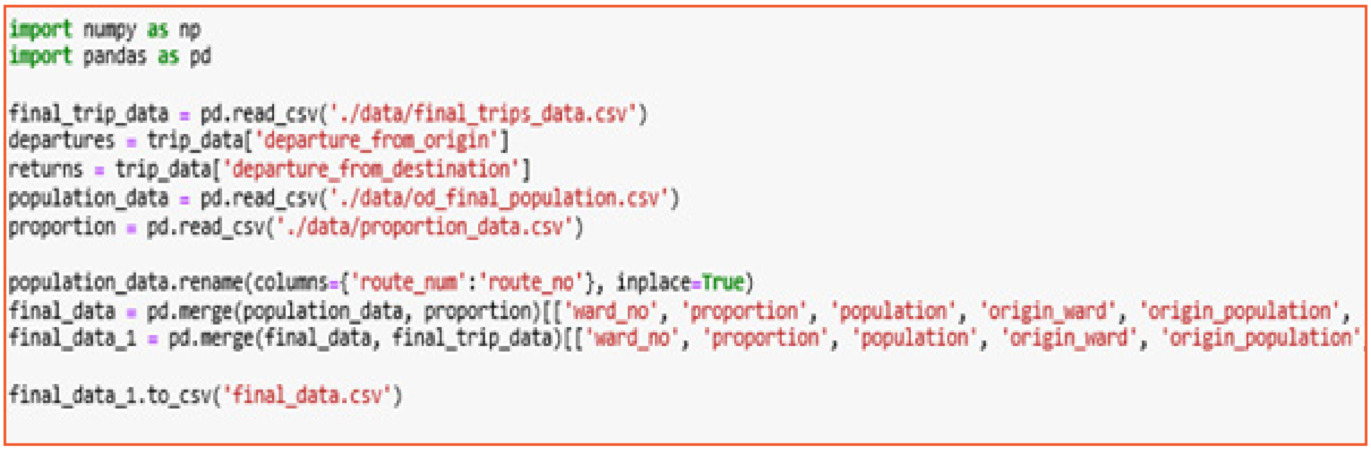
```
import pandas as pd
import json
import itertools

population_data = pd.read_csv('./data/ward_population.csv')
population_data = population_data.sort_values(by ='ward_no', ascending=True)
population_data = population_data.reset_index(drop=True)
print(population_data)

od_data = pd.read_csv("./data/od_final_grouped.csv")

convert_dict = {'ward_no': int,'origin_ward': int,'route_counts': int}

df = od_data.astype(convert_dict)
df3 = pd.merge(df,population_data)[['ward_no', 'population', 'origin_ward', 'route_num', 'route_counts']]

origin_wards = df3['origin_ward']

origin_population = []

for m in range(0, len(origin_wards)):
    origin_ward = origin_wards[m]
    ward_index = int(origin_ward) - 1
    pop = population_data.at[ward_index, 'population']

    origin_population.append(pop)

df3.insert(3, "origin_population", origin_population, True)
print(df3)
df3.to_csv('od_final_population.csv', index=False)
print(df3.count())
```

We read a csv which contains the population of the wards which I have got from here , and we convert some of the columns ‘ward_no’, ‘origin_ward’ and ‘route_counts’ to an integer as they are floats(decimals, we call them floats in programming languages). You also might be wondering what I meant by route counts, they are nothing but the total number of od_combinations.

We then continue to merge the columns of df and the population data and then calculate the population for the origin_wards.

“Merging” two datasets is the process of bringing two datasets together into one, and aligning the rows from each based on common attributes or columns and insert the column we just calculated which is ‘origin_population’ to the merged data and store it as a csv.

Next cell

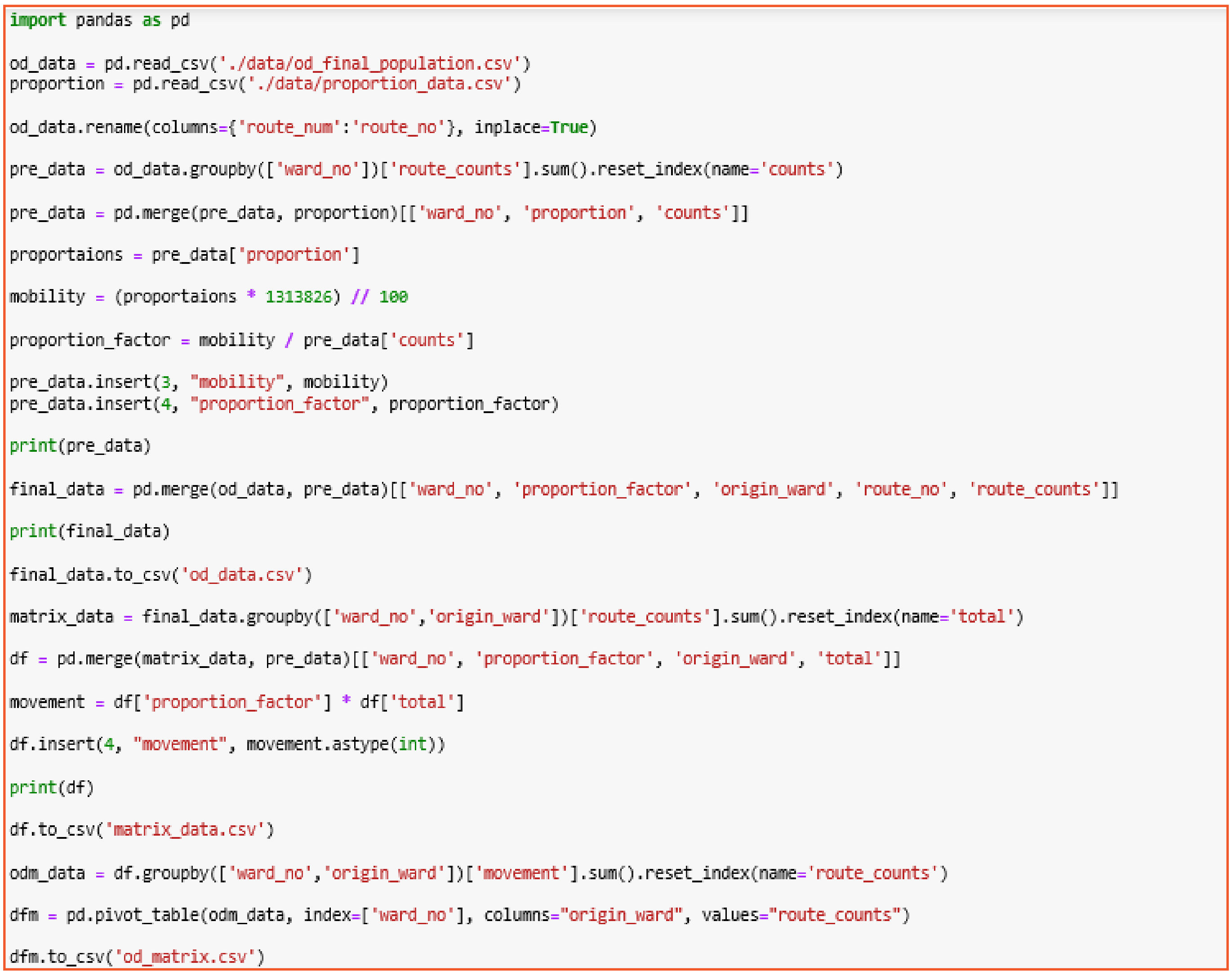


```
import numpy as np
import pandas as pd

final_trip_data = pd.read_csv('./data/final_trips_data.csv')
departures = trip_data['departure_from_origin']
returns = trip_data['departure_from_destination']
population_data = pd.read_csv('./data/od_final_population.csv')
proportion = pd.read_csv('./data/proportion_data.csv')

population_data.rename(columns={'route_num':'route_no'}, inplace=True)
final_data = pd.merge(population_data, proportion)[['ward_no', 'proportion', 'population', 'origin_ward', 'origin_population',
final_data_1 = pd.merge(final_data, final_trip_data)[['ward_no', 'proportion', 'population', 'origin_ward', 'origin_population'

final_data_1.to_csv('final_data.csv')
```

Note – I have used some files which I have generated in some file like ‘proportion_data’; etc: - in another notebook of mine which I have published in git hub as well.

This consists of the merged data of the two files I had generated in the other notebook I had mentioned before. I have just generated a csv which consists of all the data together

Next cell



```
import pandas as pd

od_data = pd.read_csv('./data/od_final_population.csv')
proportion = pd.read_csv('./data/proportion_data.csv')

od_data.rename(columns={'route_num':'route_no'}, inplace=True)

pre_data = od_data.groupby(['ward_no'])['route_counts'].sum().reset_index(name='counts')

pre_data = pd.merge(pre_data, proportion)[['ward_no', 'proportion', 'counts']]

proportaions = pre_data['proportion']

mobility = (proportaions * 1313826) // 100

proportion_factor = mobility / pre_data['counts']

pre_data.insert(3, "mobility", mobility)
pre_data.insert(4, "proportion_factor", proportion_factor)

print(pre_data)

final_data = pd.merge(od_data, pre_data)[['ward_no', 'proportion_factor', 'origin_ward', 'route_no', 'route_counts']]

print(final_data)

final_data.to_csv('od_data.csv')

matrix_data = final_data.groupby(['ward_no','origin_ward'])['route_counts'].sum().reset_index(name='total')

df = pd.merge(matrix_data, pre_data)[['ward_no', 'proportion_factor', 'origin_ward', 'total']]

movement = df['proportion_factor'] * df['total']

df.insert(4, "movement", movement.astype(int))

print(df)

df.to_csv('matrix_data.csv')

odm_data = df.groupby(['ward_no','origin_ward'])['movement'].sum().reset_index(name='route_counts')

dfm = pd.pivot_table(odm_data, index=['ward_no'], columns="origin_ward", values="route_counts")

dfm.to_csv('od_matrix.csv')
```

Now with all the data we just collected let's generate the OD matrix, which describes people's movement in a certain area. This will be useful in other notebooks, now that we have generated the od_matrix we would need to plot the graph, this will be our final step.

Notebook – 3 ““covid19_modelling”

This notebook will mainly consist of plotting the graphs using modules such as matplotlib or plotly

Plotly

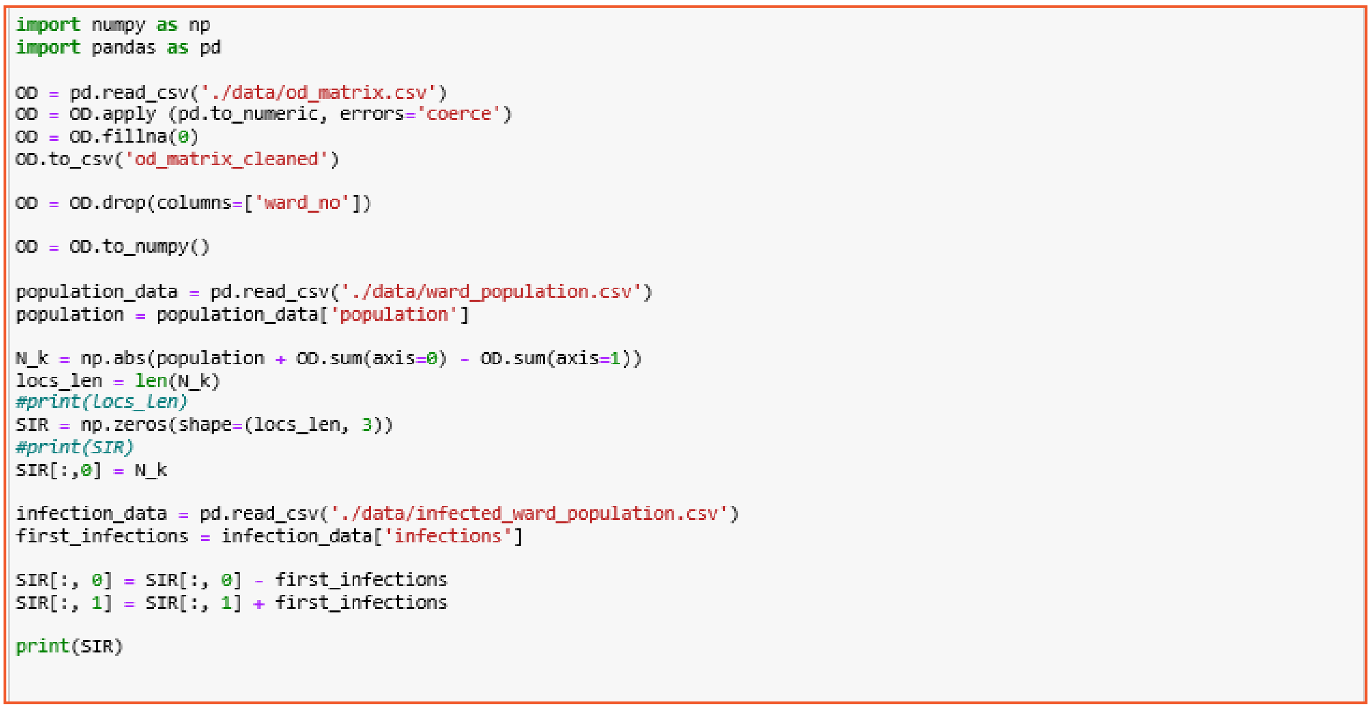
Plotly allows users to import, copy and paste, or stream data to be analyzed and visualized. For analysis and styling graphs, Plotly offers a Python sandbox (NumPy supported), datagrid, and GUI. Python scripts can be saved, shared, and collaboratively edited in Plotly.

The Plotly Python graphing library is a scientific graphing library. Graphs can be styled with Python and a GUI, and shared with a URL for others to view, collaborate, or save a copy.

Matplotlib

Matplotlib is one of the most popular Python packages used for data visualization. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

The notebook itself



```
import numpy as np
import pandas as pd

OD = pd.read_csv('./data/od_matrix.csv')
OD = OD.apply (pd.to_numeric, errors='coerce')
OD = OD.fillna(0)
OD.to_csv('od_matrix_cleaned')

OD = OD.drop(columns=['ward_no'])

OD = OD.to_numpy()

population_data = pd.read_csv('./data/ward_population.csv')
population = population_data['population']

N_k = np.abs(population + OD.sum(axis=0) - OD.sum(axis=1))
locs_len = len(N_k)
#print(locs_len)
SIR = np.zeros(shape=(locs_len, 3))
#print(SIR)
SIR[:,0] = N_k

infection_data = pd.read_csv('./data/infected_ward_population.csv')
first_infections = infection_data['infections']

SIR[:, 0] = SIR[:, 0] - first_infections
SIR[:, 1] = SIR[:, 1] + first_infections

print(SIR)
```

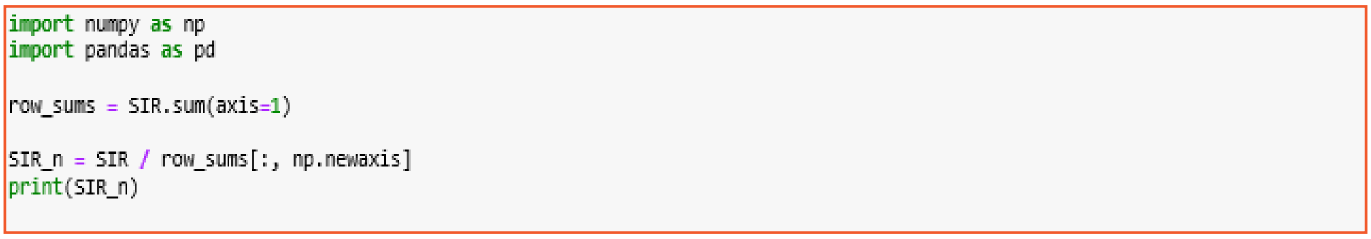
Predictive modeling uses statistics to predict outcomes. Most often the event one wants to predict is in the future, but predictive modelling can be applied to any type of unknown event, regardless of when it occurred. In my case it was predictive modelling for the future

I have used the compartmental model SIR, which stands for Susceptible, Infected and Recovered/Removed.

- Susceptible means a person who is likely to get affected by the virus,
- Infected meaning people who have Covid-19, and Recovered meaning a person who has come to a normal state of health after getting infected with
- Covid-19. The cell below calculates the SIR population from the data we have given to the code

And since we have used that model, we will have to calculate the values we require and the above cell does just, it calculates the values for the SIR using various formulas

Let’s continue.

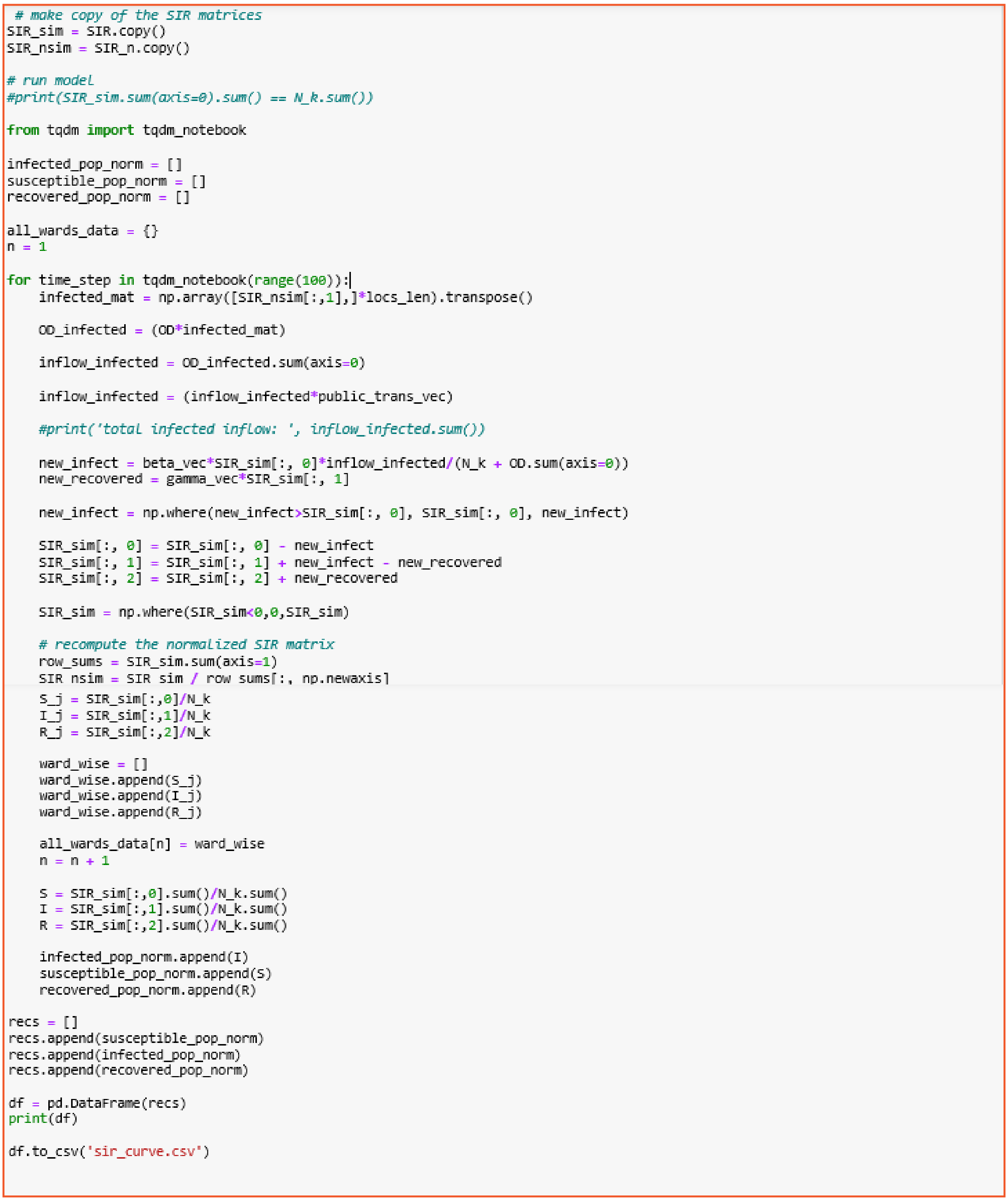


```
import numpy as np
import pandas as pd

row_sums = SIR.sum(axis=1)

SIR_n = SIR / row_sums[:, np.newaxis]
print(SIR_n)
```

This cell generates the normalized SIR with formula mentioned above, If you are wondering about what normalized means here’s the answer, (in floating-point/decimal representation) express (a number) in the standard form as regards the position of the radix point(the base of a system of numeration.), usually immediately following the first non-zero digit.

The “np. newaxis” function is used to is generally used with slicing. It indicates that you want to add an additional dimension to the array.

Let’s move on to the next cell



```
 # make copy of the SIR matrices
SIR_sim = SIR.copy()
SIR_nsim = SIR_n.copy()

# run model
#print(SIR_sim.sum(axis=0).sum() == N_k.sum())

from tqdm import tqdm_notebook

infected_pop_norm = []
susceptible_pop_norm = []
recovered_pop_norm = []

all_wards_data = {}
n = 1

for time_step in tqdm_notebook(range(100)):
    infected_mat = np.array([SIR_nsim[:,1],]*locs_len).transpose()

    OD_infected = (OD*infected_mat)

    inflow_infected = OD_infected.sum(axis=0)

    inflow_infected = (inflow_infected*public_trans_vec)

    #print('total infected inflow: ', inflow_infected.sum())

    new_infect = beta_vec*SIR_sim[:, 0]*inflow_infected/(N_k + OD.sum(axis=0))
    new_recovered = gamma_vec*SIR_sim[:, 1]

    new_infect = np.where(new_infect>SIR_sim[:, 0], SIR_sim[:, 0], new_infect)

    SIR_sim[:, 0] = SIR_sim[:, 0] - new_infect
    SIR_sim[:, 1] = SIR_sim[:, 1] + new_infect - new_recovered
    SIR_sim[:, 2] = SIR_sim[:, 2] + new_recovered

    SIR_sim = np.where(SIR_sim<0,0,SIR_sim)

    # recompute the normalized SIR matrix
    row_sums = SIR_sim.sum(axis=1)
    SIR_nsim = SIR_sim / row_sums[:, np.newaxis]
    S_j = SIR_sim[:,0]/N_k
    I_j = SIR_sim[:,1]/N_k
    R_j = SIR_sim[:,2]/N_k

    ward_wise = []
    ward_wise.append(S_j)
    ward_wise.append(I_j)
    ward_wise.append(R_j)

    all_wards_data[n] = ward_wise
    n = n + 1

    S = SIR_sim[:,0].sum()/N_k.sum()
    I = SIR_sim[:,1].sum()/N_k.sum()
    R = SIR_sim[:,2].sum()/N_k.sum()

    infected_pop_norm.append(I)
    susceptible_pop_norm.append(S)
    recovered_pop_norm.append(R)

recs = []
recs.append(susceptible_pop_norm)
recs.append(infected_pop_norm)
recs.append(recovered_pop_norm)

df = pd.DataFrame(recs)
print(df)

df.to_csv('sir_curve.csv')
```

This is the second – last step for our predictions

What does this cell do? It gets all the data for plotting the graphs, A plot is a graphical technique for representing a data set. I have also displayed the plots for every individual wards of Bangalore

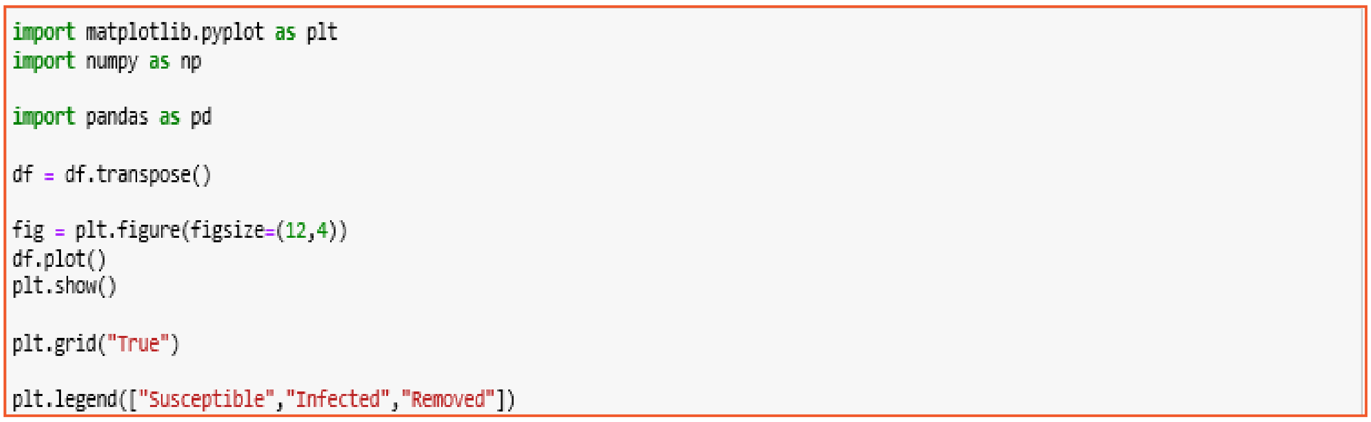
I will now show a part of my dashboard;

This going to be our final step;

Now that we have got all the data we need, let's move on to the plotting the graphs to get the SIR curve. We have used a module called matplotlib to plot the graphs you shall see below.

Great it works now we'll have to design the dashboard since people cannot go to jupyter and run this so we shall make an application (the code to do so is also given in my repository)

The code is pretty simple to understand



```
import matplotlib.pyplot as plt
import numpy as np

import pandas as pd

df = df.transpose()

fig = plt.figure(figsize=(12,4))
df.plot()
plt.show()

plt.grid("True")

plt.legend(["Susceptible","Infected","Removed"])
```

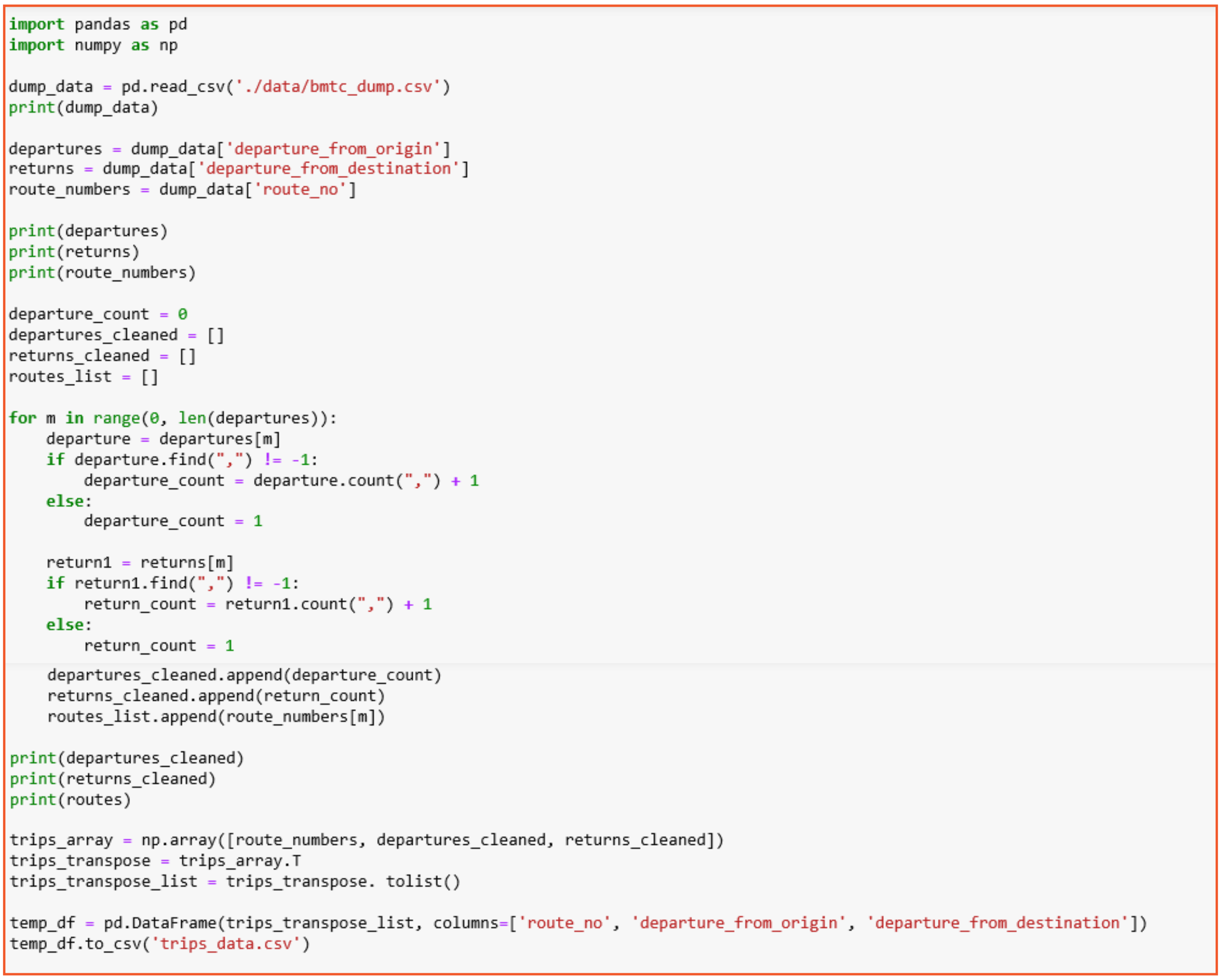
You probably guessed what this cell does, yes it plots SIR graphs "the susceptible, Infected, Removed" are the graphs values that it would display

The transpose of a matrix is obtained by moving the rows data to the column and columns data to the rows.

Final notebook – “additional_processing”

The cell below determines the total number of returns and departures for the BMTC buses (This is the data we have got from a dataset of BMTC which is The Bangalore metropolitan transport corporation.)

We also need it to determine the mobility we would need the data for the trips made and here’s how you can do it;



```python
import pandas as pd
import numpy as np

dump_data = pd.read_csv('./data/bmtc_dump.csv')
print(dump_data)

departures = dump_data['departure_from_origin']
returns = dump_data['departure_from_destination']
route_numbers = dump_data['route_no']

print(departures)
print(returns)
print(route_numbers)

departure_count = 0
departures_cleaned = []
returns_cleaned = []
routes_list = []

for m in range(0, len(departures)):
    departure = departures[m]
    if departure.find(",") != -1:
        departure_count = departure.count(",") + 1
    else:
        departure_count = 1

    return1 = returns[m]
    if return1.find(",") != -1:
        return_count = return1.count(",") + 1
    else:
        return_count = 1
    departures_cleaned.append(departure_count)
    returns_cleaned.append(return_count)
    routes_list.append(route_numbers[m])

print(departures_cleaned)
print(returns_cleaned)
print(routes)

trips_array = np.array([route_numbers, departures_cleaned, returns_cleaned])
trips_transpose = trips_array.T
trips_transpose_list = trips_transpose. tolist()

temp_df = pd.DataFrame(trips_transpose_list, columns=['route_no', 'departure_from_origin', 'departure_from_destination'])
temp_df.to_csv('trips_data.csv')
```

This cell below calculates the average for departures and returns which would then be used to determine the number of trips for buses in a single day.

```python
import numpy as np
import pandas as pd

trip_data = pd.read_csv('./data/trips_data.csv')
departures = trip_data['departure_from_origin']
returns = trip_data['departure_from_destination']
population_data = pd.read_csv('./data/od_final_population.csv')
proportion = pd.read_csv('./data/proportion_data.csv')

average = departures + returns // 2
trip_data.insert(4, "average", average)
print(trip_data)
trip_data.to_csv('final_trips_data.csv')
```

The cell below helps us determine the number of the total combinations of routes for the whole of Bangalore, we have made the column and saved it as 'route_combinations_count.csv'

```python
import pandas as pd
import json
import itertools

od_data = pd.read_csv('./data/od_final.csv')

df = od_data.groupby(['route_num'])['route_num'].count().reset_index(name='route_counts')

df.to_csv('route_combinations_count.csv')

print(df.shape)
```

Read the bmtc_dump and calculate the counts for the departures and returns.

We proceed to transpose the columns of a list and finally store them as a csv

The data we have used is the Bangalore's commercial activity data. If you're wondering why I have used this file it is because We can't assume that people would be moving around randomly and the chances of them going to work is higher than the chances of them going someplace else so that is why I have used this file and get the proportion for the values and save it as a csv which I would be using in some other cells.

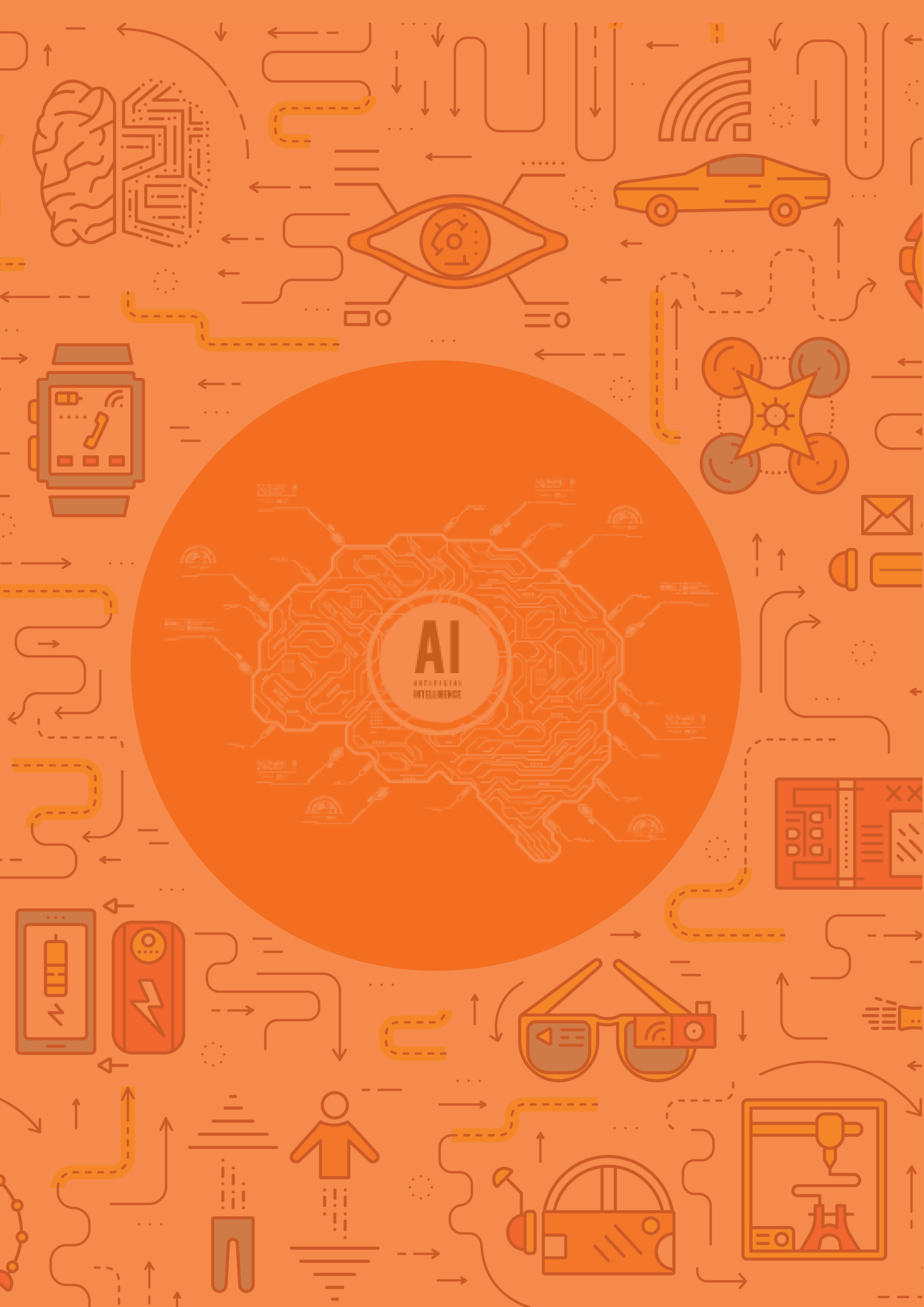
```python
import numpy as np
import pandas as pd

revenue_data = pd.read_csv('./data/ward_revenue.csv')
revenue = revenue_data['revenue']

x = revenue.sum()
proportion = revenue/x*100

revenue_data.insert(3, "proportion", proportion)

revenue_data.to_csv('proportion_data.csv')
print(revenue_data)
```

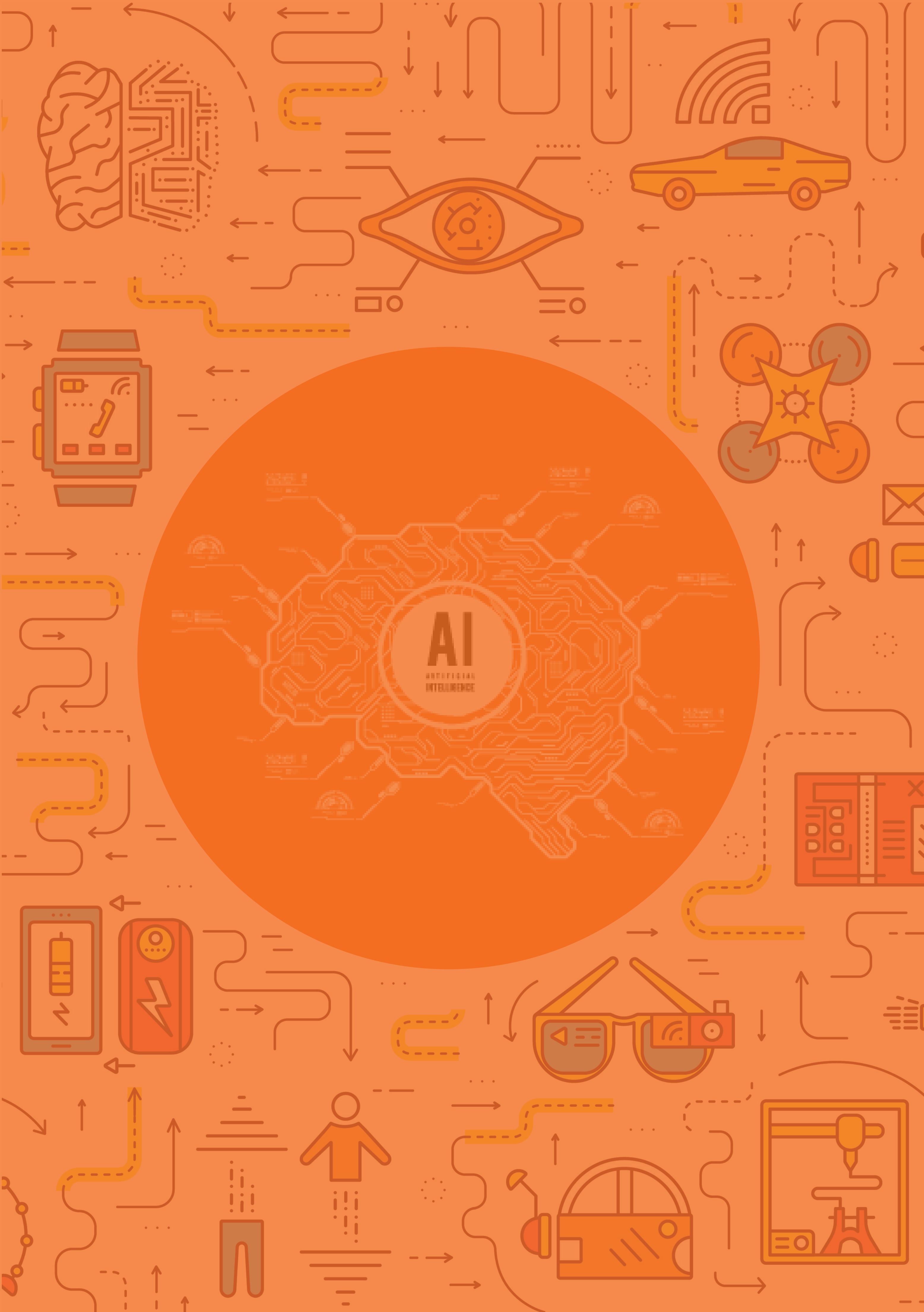
The cell below calculates the route_counts for origin_wards as that will also be necessary to determine the returns and departures

```
import pandas as pd
import json
import itertools

od_data = pd.read_csv('./data/od_final.csv')

df = od_data.groupby(['destination_ward','origin_ward'])['route_num'].count().reset_index(name='route_counts')

df = df.rename(columns={'destination_ward': 'ward_no'})

print(df)

df.to_csv('origin_wards.csv')
```

Some more projects

Realtime analytics for ride sharing workshop	https://streaming-analytics.labgui.de/
Build, train, debug, deploy and monitor with Sagemaker	https://github.com/aws-samples/reinvent2019-aim362-sagemaker-debugger-model-monitor
Sagemaker for Battlesnake AI	https://github.com/awslabs/sagemaker-battlesnake-ai


AI
ARTIFICIAL
INTELLIGENCE


AI
ARTIFICIAL
INTELLIGENCE

HAPPY
TINKERING