**IMPLEMENTATION:**

Screen Recordings – Section 14:

Screen recordings of the final project can be found here (each link is of a different device/web browser):

<https://youtu.be/KS7LGGbkwN8>

<https://youtu.be/_MGe1MHYkxs>

<https://youtu.be/5FhfaQk-48o>

The commented code is within the next 2 sections. Where an objective is being met, it is highlighted in green.

Images for the status updates page were taken from the following website: <https://www.london-tube-map.info>

This image shows all the files and folders used for the website:

Graphical user interface, text, application

Description automatically generated

HTML & CSS Files – Section 15:

index.html:

<html><!-- Root element -->

<head><!-- Head element for metadata of the webpage -->

<link rel=**"stylesheet"** type=**"text/css"** href=**"/styles.css"**><!-- References styling document called "styles.css" -->

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**><!-- Changes the view of the webpage for the user so that it matches the width of the page and initial zoom level -->

</head><!-- Closes head element for metadata of the webpage -->

<div class=**"navbar"**><!-- Defines the navbar section of the webpage -->

<a href=**"/journeyplanner.html"**class=**"left"**>**Journey Planner**</a><!-- Creates the first part of the navbar which is the Journey Planner page - includes the link to get to that webpage -->

<a href=**"/map.html"**class=**"center"**>**Map**</a><!-- Creates the second part of the navbar which is the Map page - includes the link to get to that webpage -->

<a href=**"/updates.html"**class=**"center"**>**Status Updates**</a><!-- Creates the third part of the navbar which is the Status Updates page - includes the link to get to that webpage -->

<a href=**"/arrivaltimes.html"**>**Arrival Times**</a><!-- Creates the fourth part of the navbar which is the Arrival Times page - includes the link to get to that webpage -->

<a href=**"/liftupdates.html"**>**Lift Updates**</a><!-- Creates the fifth part of the navbar which is the Lift Updates page - includes the link to get to that webpage -->

<a href=**"/index.html"**class=**"right"**>**About**</a><!-- Creates the last part of the navbar which is the About page - includes the link to get to that webpage -->

</div><!-- Closing tag for defining the navbar section of the webpage -->

<div class=**"header"**><!-- Defines the header section of the webpage -->

<h3>**Tube Journey Planner**</h3><!-- Defines a heading -->

<h3>**Farheen Fahim**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the header section of the webpage -->

<style> **/\* Defines the styling of the webpage \*/**

**div.responsive{ /\* Defines the responsive div tag \*/**

**max-width: 100%;**

**height: auto;**

**display: block;**

**margin-left: auto;**

**margin-right: auto;**

**}/\* Closes the responsive div tag \*/**

</style><!-- Closes the styling element of the webpage -->

<body><!-- Defines the body of the webpage -->

<p><!-- Defines a new paragraph -->

**The London Underground (otherwise known as "The Tube") is a transportation service used by approximately 2 million people per day.**

**The system has been used since 1863. It started with only 1 line (the Metropolitan Line).**

**159 years later and the service now has 11 underground lines, 272 stations, and covers 420 km of the Capital.**

**To people who do not live in London or are not familiar with London, travel and the tube can seem daunting;**

**you could be accidentally getting on a train to the wrong place,**

**or you might not know how to navigate around London by using a tube map because it is so overwhelming.**

</p><!-- Closing tag for defining a new paragraph -->

<br><!-- Defines a line break on the webpage -->

<img src=**"https://ichef.bbci.co.uk/images/ic/896x504/p01l8c6d.jpg"**class= **"responsive"**><!-- Defines an image using a link and makes it responsive so that it adjusts to the window size -->

<p><!-- Defines a new paragraph -->

**This website aims to help people navigate London via the tube.**

**You can input your start station, end station, your walking speed, and if you are travelling during rush hour or the weekend.**

**Then, your route will be displayed on the screen, along with the corresponding lines you should take,**

**And the approximated time taken to reach there.**

</p><!-- Closing tag for defining a new paragraph -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<div class=**"footer"**><!-- Defines the footer section of the webpage -->

<h3>**Computer Science: Non-Exam Assessment**</h3><!-- Defines a heading -->

<h3>**2022/23**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the footer section of the webpage -->

</body><!-- Closing tag for defining the body of the webpage -->

A screenshot of a computer

Description automatically generated</html><!-- Closing tag for root element -->

journeyplanner.html:

<html><!-- Root element -->

<head><!-- Head element for metadata of the webpage -->

<title><!-- Title element -->

**A Journey Planner** <!-- Defines the title of the webpage -->

</title><!-- Closes title element -->

<link rel=**"stylesheet"** type=**"text/css"** href=**"styles.css"**><!-- References styling document called "styles.css" -->

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**><!-- Changes the view of the webpage for the user so that it matches the width of the page and initial zoom level -->

</head><!-- Closes head element for metadata of the webpage -->

<div class=**"navbar"**><!-- Defines the navbar section of the webpage -->

<a href=**"/journeyplanner.html"**class=**"left"**>**Journey Planner**</a><!-- Creates the first part of the navbar which is the Journey Planner page - includes the link to get to that webpage -->

<a href=**"/map.html"**class=**"center"**>**Map**</a><!-- Creates the second part of the navbar which is the Map page - includes the link to get to that webpage -->

<a href=**"/updates.html"**class=**"center"**>**Status Updates**</a><!-- Creates the third part of the navbar which is the Status Updates page - includes the link to get to that webpage -->

<a href=**"/arrivaltimes.html"**>**Arrival Times**</a><!-- Creates the fourth part of the navbar which is the Arrival Times page - includes the link to get to that webpage -->

<a href=**"/liftupdates.html"**>**Lift Updates**</a><!-- Creates the fifth part of the navbar which is the Lift Updates page - includes the link to get to that webpage -->

<a href=**"/index.html"**class=**"right"**>**About**</a><!-- Creates the last part of the navbar which is the About page - includes the link to get to that webpage -->

</div><!-- Closing tag for defining the navbar section of the webpage -->

<div class=**"header"**><!-- Defines the header section of the webpage -->

<h3>**Tube Journey Planner**</h3><!-- Defines a heading -->

<h3>**Farheen Fahim**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the header section of the webpage -->

<body><!-- Defines the body of the webpage -->

<h1>**JOURNEY PLANNER**</h1><!-- Defines a heading -->

<p><!-- Defines a new paragraph -->

<form method=**"POST"** action=**"/cgi-bin/journeyplanner.pyw"**><!-- Defines the form which the user needs to input values into and submit, also defines the action for when the form is submitted -->

<!-- Objectives 1.1, 4.3, -->

**Start Station:**

<select name=**"station1"**><!-- Defines a select element which allows the user to choose an input from a pre-determined selection of values -->

<!-- Defines option tags which give the user options for their input -->

<option value=**"Start Station"**>**Start Station**</option>

<option value=**"Aldgate"**>**Aldgate**</option>

<option value=**"Aldgate East"**>**Aldgate East (Step-Free)**</option>

<option value=**"Angel"**>**Angel**</option>

<option value=**"Baker Street"**>**Baker Street (Step-Free)**</option>

<option value=**"Bank"**>**Bank**</option>

<option value=**"Barbican"**>**Barbican (Step-Free)**</option>

<option value=**"Battersea Power Station"**>**Battersea Power Station (Step-Free)**</option>

<option value=**"Bayswater"**>**Bayswater**</option>

<option value=**"Blackfriars"**>**Blackfriars (Step-Free)**</option>

<option value=**"Bond Street"**>**Bond Street (Step-Free)**</option>

<option value=**"Borough"**>**Borough (Step-Free)**</option>

<option value=**"Cannon Street"**>**Cannon Street (Step-Free)**</option>

<option value=**"Chancery Lane"**>**Chancery Lane**</option>

<option value=**"Charing Cross"**>**Charing Cross**</option>

<option value=**"Covent Garden"**>**Covent Garden**</option>

<option value=**"Earl's Court"**>**Earl's Court (Step-Free)**</option>

<option value=**"Edgware Road"**>**Edgware Road (Step-Free)**</option>

<option value=**"Elephant & Castle"**>**Elephant** & **Castle (Step-Free)**</option>

<option value=**"Embankment"**>**Embankment**</option>

<option value=**"Euston"**>**Euston (Step-Free)**</option>

<option value=**"Euston Square"**>**Euston Square (Step-Free)**</option>

<option value=**"Farringdon"**>**Farringdon (Step-Free)**</option>

<option value=**"Gloucester Road"**>**Gloucester Road**</option>

<option value=**"Goodge Street"**>**Goodge Street**</option>

<option value=**"Great Portland Street"**>**Great Portland Street**</option>

<option value=**"Green Park"**>**Green Park (Step-Free)**</option>

<option value=**"High Street Kensington"**>**High Street Kensington**</option>

<option value=**"Holborn"**>**Holborn**</option>

<option value=**"Hyde Park Corner"**>**Hyde Park Corner**</option>

<option value=**"Kennington"**>**Kennington (Step-Free)**</option>

<option value=**"King's Cross St. Pancras"**>**King's Cross St. Pancras (Step-Free)option>**

<option value=**"Knightsbridge"**>**Knightsbridge**</option>

<option value=**"Lambeth North"**>**Lambeth North**</option>

<option value=**"Lancaster Gate"**>**Lancaster Gate**</option>

<option value=**"Leicester Square"**>**Leicester Square**</option>

<option value=**"Liverpool Street"**>**Liverpool Street (Step-Free)**</option>

<option value=**"London Bridge"**>**London Bridge (Step-Free)**</option>

<option value=**"Mansion House"**>**Mansion House**</option>

<option value=**"Marble Arch"**>**Marble Arch**</option>

<option value=**"Marylebone"**>**Marylebone**</option>

<option value=**"Moorgate"**>**Moorgate (Step-Free)**</option>

<option value=**"Monument"**>**Monument**</option>

<option value=**"Nine Elms"**>**Nine Elms (Step-Free)**</option>

<option value=**"Notting Hill Gate"**>**Notting Hill Gate**</option>

<option value=**"Old Street"**>**Old Street**</option>

<option value=**"Oxford Circus"**>**Oxford Circus (Step-Free)**</option>

<option value=**"Paddington"**>**Paddington (Step-Free)**</option>

<option value=**"Piccadilly Circus"**>**Piccadilly Circus**</option>

<option value=**"Pimlico"**>**Pimlico**</option>

<option value=**"Queensway"**>**Queensway**</option>

<option value=**"Regent's Park"**>**Regent's Park**</option>

<option value=**"Russell Square"**>**Russell Square**</option>

<option value=**"Sloane Square"**>**Sloane Square**</option>

<option value=**"South Kensington"**>**South Kensington (Step-Free)**</option>

<option value=**"Southwark"**>**Southwark (Step-Free)**</option>

<option value=**"St. James's Park"**>**St. James's Park**</option>

<option value=**"St. Paul's"**>**St. Paul's**</option>

<option value=**"Temple"**>**Temple**</option>

<option value=**"Tottenham Court Road"**>**Tottenham Court Road (Step-Free)**</option>

<option value=**"Tower Hill"**>**Tower Hill (Step-Free)**</option>

<option value=**"Vauxhall"**>**Vauxhall (Step-Free)**</option>

<option value=**"Victoria"**>**Victoria (Step-Free)**</option>

<option value=**"Warren Street"**>**Warren Street**</option>

<option value=**"Waterloo"**>**Waterloo (Step-Free)**</option>

<option value=**"Westminster"**>**Westminster (Step-Free)**</option>

</select><!-- Closing tag for the select element -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

**End Station:**

<select name=**"station2"**><!-- Defines a select element which allows the user to choose an input from a pre-determined selection of values -->

<!-- Defines option tags which give the user options for their input -->

<option value=**"Start Station"**>**Start Station**</option>

<option value=**"Aldgate"**>**Aldgate**</option>

<option value=**"Aldgate East"**>**Aldgate East (Step-Free)**</option>

<option value=**"Angel"**>**Angel**</option>

<option value=**"Baker Street"**>**Baker Street (Step-Free)**</option>

<option value=**"Bank"**>**Bank**</option>

<option value=**"Barbican"**>**Barbican (Step-Free)**</option>

<option value=**"Battersea Power Station"**>**Battersea Power Station (Step-Free)**</option>

<option value=**"Bayswater"**>**Bayswater**</option>

<option value=**"Blackfriars"**>**Blackfriars (Step-Free)**</option>

<option value=**"Bond Street"**>**Bond Street (Step-Free)**</option>

<option value=**"Borough"**>**Borough (Step-Free)**</option>

<option value=**"Cannon Street"**>**Cannon Street (Step-Free)**</option>

<option value=**"Chancery Lane"**>**Chancery Lane**</option>

<option value=**"Charing Cross"**>**Charing Cross**</option>

<option value=**"Covent Garden"**>**Covent Garden**</option>

<option value=**"Earl's Court"**>**Earl's Court (Step-Free)**</option>

<option value=**"Edgware Road"**>**Edgware Road (Step-Free)**</option>

<option value=**"Elephant & Castle"**>**Elephant** & **Castle (Step-Free)**</option>

<option value=**"Embankment"**>**Embankment**</option>

<option value=**"Euston"**>**Euston (Step-Free)**</option>

<option value=**"Euston Square"**>**Euston Square (Step-Free)**</option>

<option value=**"Farringdon"**>**Farringdon (Step-Free)**</option>

<option value=**"Gloucester Road"**>**Gloucester Road**</option>

<option value=**"Goodge Street"**>**Goodge Street**</option>

<option value=**"Great Portland Street"**>**Great Portland Street**</option>

<option value=**"Green Park"**>**Green Park (Step-Free)**</option>

<option value=**"High Street Kensington"**>**High Street Kensington**</option>

<option value=**"Holborn"**>**Holborn**</option>

<option value=**"Hyde Park Corner"**>**Hyde Park Corner**</option>

<option value=**"Kennington"**>**Kennington (Step-Free)**</option>

<option value=**"King's Cross St. Pancras"**>**King's Cross St. Pancras (Step-Free)option>**

<option value=**"Knightsbridge"**>**Knightsbridge**</option>

<option value=**"Lambeth North"**>**Lambeth North**</option>

<option value=**"Lancaster Gate"**>**Lancaster Gate**</option>

<option value=**"Leicester Square"**>**Leicester Square**</option>

<option value=**"Liverpool Street"**>**Liverpool Street (Step-Free)**</option>

<option value=**"London Bridge"**>**London Bridge (Step-Free)**</option>

<option value=**"Mansion House"**>**Mansion House**</option>

<option value=**"Marble Arch"**>**Marble Arch**</option>

<option value=**"Marylebone"**>**Marylebone**</option>

<option value=**"Moorgate"**>**Moorgate (Step-Free)**</option>

<option value=**"Monument"**>**Monument**</option>

<option value=**"Nine Elms"**>**Nine Elms (Step-Free)**</option>

<option value=**"Notting Hill Gate"**>**Notting Hill Gate**</option>

<option value=**"Old Street"**>**Old Street**</option>

<option value=**"Oxford Circus"**>**Oxford Circus (Step-Free)**</option>

<option value=**"Paddington"**>**Paddington (Step-Free)**</option>

<option value=**"Piccadilly Circus"**>**Piccadilly Circus**</option>

<option value=**"Pimlico"**>**Pimlico**</option>

<option value=**"Queensway"**>**Queensway**</option>

<option value=**"Regent's Park"**>**Regent's Park**</option>

<option value=**"Russell Square"**>**Russell Square**</option>

<option value=**"Sloane Square"**>**Sloane Square**</option>

<option value=**"South Kensington"**>**South Kensington (Step-Free)**</option>

<option value=**"Southwark"**>**Southwark (Step-Free)**</option>

<option value=**"St. James's Park"**>**St. James's Park**</option>

<option value=**"St. Paul's"**>**St. Paul's**</option>

<option value=**"Temple"**>**Temple**</option>

<option value=**"Tottenham Court Road"**>**Tottenham Court Road (Step-Free)**</option>

<option value=**"Tower Hill"**>**Tower Hill (Step-Free)**</option>

<option value=**"Vauxhall"**>**Vauxhall (Step-Free)**</option>

<option value=**"Victoria"**>**Victoria (Step-Free)**</option>

<option value=**"Warren Street"**>**Warren Street**</option>

<option value=**"Waterloo"**>**Waterloo (Step-Free)**</option>

<option value=**"Westminster"**>**Westminster (Step-Free)**</option>

</select><!-- Closing tag for the select element -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<!-- Objective 5: -->

**Rush Hour?**

<select name=**"rush\_hour"**><!-- Defines a select element which allows the user to choose an input from a pre-determined selection of values -->

<!-- Defines option tags which give the user options for their input -->

<option value=**"No"**>**No**</option>

<option value=**"Yes"**>**Yes**</option>

</select><!-- Closing tag for the select element -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

**Weekend?**

<select name=**"weekend"**><!-- Defines a select element which allows the user to choose an input from a pre-determined selection of values -->

<!-- Defines option tags which give the user options for their input -->

<option value=**"No"**>**No**</option>

<option value=**"Yes"**>**Yes**</option>

</select><!-- Closing tag for the select element -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<!-- Objective 7.1: -->

**Step Free?**

<select name=**"stepfree"**><!-- Defines a select element which allows the user to choose an input from a pre-determined selection of values -->

<!-- Defines option tags which give the user options for their input -->

<option value=**"No"**>**No**</option>

<option value=**"Yes"**>**Yes**</option>

</select><!-- Closing tag for the select element -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<input type=**"submit"** value=**"Find Route"**><!-- Defines an input tag which involves the user pressing and making the form submit -->

</form><!-- Closes the tag defining the form which the user needs to input values into and submit -->

</p><!-- Closing tag for defining a new paragraph -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<div class=**"footer"**><!-- Defines the footer section of the webpage -->

<h3>**Computer Science: Non-Exam Assessment**</h3><!-- Defines a heading -->

<h3>**2022/23**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the footer section of the webpage -->

</body><!-- Closing tag for defining the body of the webpage -->

A screenshot of a computer

Description automatically generated</html><!-- Closing tag for root element -->

map.html:

<html><!-- Root element -->

<head><!-- Head element for metadata of the webpage -->

<link rel=**"stylesheet"** type=**"text/css"** href=**"/styles.css"**><!-- References styling document called "styles.css" -->

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**><!-- Changes the view of the webpage for the user so that it matches the width of the page and initial zoom level -->

</head><!-- Closes head element for metadata of the webpage -->

<title><!-- Title element -->

**Tube Map** <!-- Defines the title of the webpage -->

</title><!-- Closes title element -->

<div class=**"navbar"**><!-- Defines the navbar section of the webpage -->

<a href=**"/journeyplanner.html"**class=**"left"**>**Journey Planner**</a><!-- Creates the first part of the navbar which is the Journey Planner page - includes the link to get to that webpage -->

<a href=**"/map.html"**class=**"center"**>**Map**</a><!-- Creates the second part of the navbar which is the Map page - includes the link to get to that webpage -->

<a href=**"/updates.html"**class=**"center"**>**Status Updates**</a><!-- Creates the third part of the navbar which is the Status Updates page - includes the link to get to that webpage -->

<a href=**"/arrivaltimes.html"**>**Arrival Times**</a><!-- Creates the fourth part of the navbar which is the Arrival Times page - includes the link to get to that webpage -->

<a href=**"/liftupdates.html"**>**Lift Updates**</a><!-- Creates the fifth part of the navbar which is the Lift Updates page - includes the link to get to that webpage -->

<a href=**"/index.html"**class=**"right"**>**About**</a><!-- Creates the last part of the navbar which is the About page - includes the link to get to that webpage -->

</div><!-- Closing tag for defining the navbar section of the webpage -->

<div class=**"header"**><!-- Defines the header section of the webpage -->

<h3>**Tube Journey Planner**</h3><!-- Defines a heading -->

<h3>**Farheen Fahim**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the header section of the webpage -->

<body><!-- Defines the body of the webpage -->

<img src=**"https://tfl.gov.uk/cdn/static/cms/images/tube-map.gif"**class= **"responsive"**><!-- Defines an image using a link and makes it responsive so that it adjusts to the window size -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<div class=**"footer"**><!-- Defines the footer section of the webpage -->

<h3>**Computer Science: Non-Exam Assessment**</h3><!-- Defines a heading -->

<h3>**2022/23**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the footer section of the webpage -->

</body><!-- Closing tag for defining the body of the webpage -->

</html><!-- Closing tag for root element -->

A screenshot of a computer

Description automatically generated

updates.html:

<html><!-- Root element -->

<head><!-- Head element for metadata of the webpage -->

<title><!-- Title element -->

**Status Updates For London Tube** <!-- Defines the title of the webpage -->

</title><!-- Closes title element -->

<link rel=**"stylesheet"** type=**"text/css"** href=**"/styles.css"**><!-- References styling document called "styles.css" -->

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**><!-- Changes the view of the webpage for the user so that it matches the width of the page and initial zoom level -->

</head><!-- Closes head element for metadata of the webpage -->

<div class=**"navbar"**><!-- Defines the navbar section of the webpage -->

<a href=**"/journeyplanner.html"**class=**"left"**>**Journey Planner**</a><!-- Creates the first part of the navbar which is the Journey Planner page - includes the link to get to that webpage -->

<a href=**"/map.html"**class=**"center"**>**Map**</a><!-- Creates the second part of the navbar which is the Map page - includes the link to get to that webpage -->

<a href=**"/updates.html"**class=**"center"**>**Status Updates**</a><!-- Creates the third part of the navbar which is the Status Updates page - includes the link to get to that webpage -->

<a href=**"/arrivaltimes.html"**>**Arrival Times**</a><!-- Creates the fourth part of the navbar which is the Arrival Times page - includes the link to get to that webpage -->

<a href=**"/liftupdates.html"**>**Lift Updates**</a><!-- Creates the fifth part of the navbar which is the Lift Updates page - includes the link to get to that webpage -->

<a href=**"/index.html"**class=**"right"**>**About**</a><!-- Creates the last part of the navbar which is the About page - includes the link to get to that webpage -->

</div><!-- Closing tag for defining the navbar section of the webpage -->

<div class=**"header"**><!-- Defines the header section of the webpage -->

<h3>**Tube Journey Planner**</h3><!-- Defines a heading -->

<h3>**Farheen Fahim**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the header section of the webpage -->

<body><!-- Defines the body of the webpage -->

<h1>**STATUS UPDATES**</h1><!-- Defines a heading -->

<p><!-- Defines a new paragraph -->

<form method=**"POST"** action=**"/cgi-bin/status\_updates.pyw"**><!-- Defines the form which the user needs to input values into and submit, also defines the action for when the form is submitted -->

**Line:**

<select name=**"line"**><!-- Defines a select element which allows the user to choose an input from a pre-determined selection of values -->

<!-- Defines option tags which give the user options for their input -->

<option value = **"Choose Line"**>**Choose Line**</option>

<option value = **"Bakerloo"**>**Bakerloo**</option>

<option value = **"Central"**>**Central**</option>

<option value = **"Circle"**>**Circle**</option>

<option value = **"District"**>**District**</option>

<option value = **"Hammersmith & City"**>**Hammersmith** & **City**</option>

<option value = **"Jubilee"**>**Jubilee**</option>

<option value = **"Metropolitan"**>**Metropolitan**</option>

<option value = **"Northern"**>**Northern**</option>

<option value = **"Piccadilly"**>**Piccadilly**</option>

<option value = **"Victoria"**>**Victoria**</option>

<option value = **"Waterloo & City"**>**Waterloo** & **City**</option>

</select><!-- Closing tag for the select element -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<input type=**"submit"** value=**"Check Status"**><!-- Defines an input tag which involves the user pressing and making the form submit -->

</form><!-- Closes the tag defining the form which the user needs to input values into and submit -->

</p><!-- Closing tag for defining a new paragraph -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<div class=**"footer"**><!-- Defines the footer section of the webpage -->

<h3>**Computer Science: Non-Exam Assessment**</h3><!-- Defines a heading -->

<h3>**2022/23**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the footer section of the webpage -->

</body><!-- Closing tag for defining the body of the webpage -->

**A screenshot of a computer

Description automatically generated** </html><!-- Closing tag for root element -->

arrivaltimes.html:

<html><!-- Root element -->

<head><!-- Head element for metadata of the webpage -->

<title><!-- Title element -->

**Arrival Times** <!-- Defines the title of the webpage -->

</title><!-- Closes title element -->

<link rel=**"stylesheet"** type=**"text/css"** href=**"styles.css"**><!-- References styling document called "styles.css" -->

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**><!-- Changes the view of the webpage for the user so that it matches the width of the page and initial zoom level -->

</head><!-- Closes head element for metadata of the webpage -->

<div class=**"navbar"**><!-- Defines the navbar section of the webpage -->

<a href=**"/journeyplanner.html"**class=**"left"**>**Journey Planner**</a><!-- Creates the first part of the navbar which is the Journey Planner page - includes the link to get to that webpage -->

<a href=**"/map.html"**class=**"center"**>**Map**</a><!-- Creates the second part of the navbar which is the Map page - includes the link to get to that webpage -->

<a href=**"/updates.html"**class=**"center"**>**Status Updates**</a><!-- Creates the third part of the navbar which is the Status Updates page - includes the link to get to that webpage -->

<a href=**"/arrivaltimes.html"**>**Arrival Times**</a><!-- Creates the fourth part of the navbar which is the Arrival Times page - includes the link to get to that webpage -->

<a href=**"/liftupdates.html"**>**Lift Updates**</a><!-- Creates the fifth part of the navbar which is the Lift Updates page - includes the link to get to that webpage -->

<a href=**"/index.html"**class=**"right"**>**About**</a><!-- Creates the last part of the navbar which is the About page - includes the link to get to that webpage -->

</div><!-- Closing tag for defining the navbar section of the webpage -->

<div class=**"header"**><!-- Defines the header section of the webpage -->

<h3>**Tube Journey Planner**</h3><!-- Defines a heading -->

<h3>**Farheen Fahim**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the header section of the webpage -->

<body><!-- Defines the body of the webpage -->

<h1>**ARRIVAL TIMES**</h1><!-- Defines a heading -->

<p><!-- Defines a new paragraph -->

<form method=**"POST"** action=**"/cgi-bin/arrivaltimes.pyw"**><!-- Defines the form which the user needs to input values into and submit, also defines the action for when the form is submitted -->

**Station:** <input type=**"text"** name=**"station\_arrivals"**><!-- Defines an input tag so that the user can enter any input without restrictions -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<input type=**"submit"** value=**"Find Arrival Times"**><!-- Defines an input tag which involves the user pressing and making the form submit -->

</form><!-- Closes the tag defining the form which the user needs to input values into and submit -->

</p><!-- Closing tag for defining a new paragraph -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<div class=**"footer"**><!-- Defines the footer section of the webpage -->

<h3>**Computer Science: Non-Exam Assessment**</h3><!-- Defines a heading -->

<h3>**2022/23**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the footer section of the webpage -->

</body><!-- Closing tag for defining the body of the webpage -->

</html><!-- Closing tag for root element -->

**A screenshot of a computer

Description automatically generated**

liftupdates.html:

<html><!-- Root element -->

<head><!-- Head element for metadata of the webpage -->

<title><!-- Title element -->

**Lift Updates**<!-- Defines the title of the webpage -->

</title><!-- Closes title element -->

<link rel=**"stylesheet"** type=**"text/css"** href=**"styles.css"**><!-- References styling document called "styles.css" -->

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**><!-- Changes the view of the webpage for the user so that it matches the width of the page and initial zoom level -->

</head><!-- Closes head element for metadata of the webpage -->

<div class=**"navbar"**><!-- Defines the navbar section of the webpage -->

<a href=**"/journeyplanner.html"**class=**"left"**>**Journey Planner**</a><!-- Creates the first part of the navbar which is the Journey Planner page - includes the link to get to that webpage -->

<a href=**"/map.html"**class=**"center"**>**Map**</a><!-- Creates the second part of the navbar which is the Map page - includes the link to get to that webpage -->

<a href=**"/updates.html"**class=**"center"**>**Status Updates**</a><!-- Creates the third part of the navbar which is the Status Updates page - includes the link to get to that webpage -->

<a href=**"/arrivaltimes.html"**>**Arrival Times**</a><!-- Creates the fourth part of the navbar which is the Arrival Times page - includes the link to get to that webpage -->

<a href=**"/liftupdates.html"**>**Lift Updates**</a><!-- Creates the fifth part of the navbar which is the Lift Updates page - includes the link to get to that webpage -->

<a href=**"/index.html"**class=**"right"**>**About**</a><!-- Creates the last part of the navbar which is the About page - includes the link to get to that webpage -->

</div><!-- Closing tag for defining the navbar section of the webpage -->

<div class=**"header"**><!-- Defines the header section of the webpage -->

<h3>**Tube Journey Planner**</h3><!-- Defines a heading -->

<h3>**Farheen Fahim**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the header section of the webpage -->

<body><!-- Defines the body of the webpage -->

<h1>**LIFT UPDATES**</h1><!-- Defines a heading -->

<p><!-- Defines a new paragraph -->

<form method=**"POST"** action=**"/cgi-bin/liftdisruptions.pyw"**><!-- Defines the form which the user needs to input values into and submit, also defines the action for when the form is submitted -->

**Station:** <input type=**"text"** name=**"station\_lifts"**><!-- Defines an input tag so that the user can enter any input without restrictions -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<input type=**"submit"** value=**"Find Lift Disruptions"**><!-- Defines an input tag which involves the user pressing and making the form submit -->

</form><!-- Closes the tag defining the form which the user needs to input values into and submit -->

</p><!-- Closing tag for defining a new paragraph -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<br><!-- Defines a line break on the webpage -->

<div class=**"footer"**><!-- Defines the footer section of the webpage -->

<h3>**Computer Science: Non-Exam Assessment**</h3><!-- Defines a heading -->

<h3>**2022/23**</h3><!-- Defines a heading -->

</div><!-- Closing tag for defining the footer section of the webpage -->

</body><!-- Closing tag for defining the body of the webpage -->

</html><!-- Closing tag for root element -->

A screenshot of a computer

Description automatically generated

styles.css:

/\* Objectives 4, 4.2, 6.2 \*/

/\* Objective 4.1 \*/

**.**header**{**

**padding: 1px;**

**background-color:rgb(0, 0, 0);**

**color:white;**

**font: size 5px;**

**font-family:**'Times New Roman'**, Times, serif;**

**text-align:center**

**}**

**.**footer**{**

**position: fixed;**

**padding: 1px;**

**background-color:rgb(0, 0, 0);**

**color:white;**

**font: size 5px;**

**font-family:**'Times New Roman'**, Times, serif;**

**text-align:center;**

**left: 0;**

**bottom: 0;**

**width: 100%;**

**}**

/\* Objective 3.1 \*/

**.**navbar**{**

**overflow: hidden;**

**padding: 20px;**

**background-color:rgb(100,100,100);**

**color:white;**

**font: size 20px;**

**font-family:**'Times New Roman'**, Times, serif;**

**}**

**.**navbar a **{**

**float: left;**

**color:white;**

**padding:20px 40px;**

**}**

**.**navbar **.**right **{**

**float: right;**

**}**

**.**navbar **.**left **{**

**float: left;**

**}**

**.**navbar **.**center **{**

**float: center;**

**}**

**.**navbar a**:hover** **{**

**background-color: rgb(255, 255, 255);**

**color: black;**

**}**

body**{**

**color:rgb(100, 100, 100);**

**text-align: center;**

**background-color:white;**

**font-family:sans-serif;}**

p**{**

**color:rgb(0, 0, 0);**

**text-align: center;**

**background-color:rgb(255, 255, 255);**

**font-family:Arial, Helvetica, sans-serif;}**

h2**{color: black;**

**}**

h1**{**

**text-decoration-line: underline;**

**}**

**.**responsive **{**

**width: 100%;**

**max-width: 400px;**

**height: auto;**

**}**

Python Files – Section 16:

webserver.py:

**from** http**.**server **import** HTTPServer**,** CGIHTTPRequestHandler # Import modules which will be used later in the program.

# HTTPServer is a class which is needed in the program.

# CGIHTTPRequestHandler is a class needed to create the handler object.

# Objective 10:

**class** **Handler(**CGIHTTPRequestHandler**):** # Creates the class called "handler" which is used when a form is submitted.

# Takes the class CGIHTTPRequestHandler as an argument so that the attributes and behaviours are inherited.

cgi\_directories **=** **[**"/cgi-bin"**]** # Sets the variable "cgi\_directories" to "/cgi-bin" which ensures that the scripts which are in the "cgi-bin" folder run when forms are submitted.

**print(**"Webserver running on port 8080"**)** # A print statement to show that the webserver is running, since nothing else will be printed in the console.

# Objective 10.1:

HTTPServer**((**""**,** 8080**),** Handler**).**serve\_forever**()**

# HTTPServer is a class which takes 2 arguments:

# Arg 1: server address: the address to listen to ("" is localhost) and the port number to listen to (80 as this is the port used for http by convention)

# Arg 2: request handler class: Handler class, which was defined earlier, is used in order to know where cgi scripts are and to handle all requests.

# .serve\_forever() ensures that, when the program is run, the web server runs constantly.

journeyplanner.pyw:

**import** cgi**,** cgitb**,** math # Import modules which will be used later in the program.

form **=** cgi**.**FieldStorage**()** # Data inputted from the form is stored so that it can be used later in the program.

# Objective 1:

station1**=**form**.**getvalue**(**'station1'**)** # The start station (that the user inputted) is accessed from the form and stored as the variable called "station1".

station2**=**form**.**getvalue**(**'station2'**)** # The end station (that the user inputted) is accessed from the form and stored as the variable called "station2".

rush\_hour**=**form**.**getvalue**(**'rush\_hour'**)** # The value for rush hour (that the user inputted) is accessed from the form and stored as the variable called "rush\_hour".

weekend**=**form**.**getvalue**(**'weekend'**)** # The value for if it is the weekend (that the user inputted) is accessed from the form and stored as the variable called "weekend".

step\_free**=**form**.**getvalue**(**'stepfree'**)** # The value for if the journey is step-free (that the user inputted) is accessed from the form and stored as the variable called "stepfree".

# Print statements output as html lines:

**print** **(**"Content-type:text/html\n\n"**)** # html line: defines the content type so that the lines are processed and interpretted correctly by the computer.

**print** **(**"<html>"**)** # html line: the <html> element for structuring the web page.

**print** **(**"<head>"**)** # html line: the <head> element for structuring the web page.

**print** **(**"<title>Arrivals At Tube Stations</title>"**)** # html line: the <title> element for distinguishing which page the user is on.

**print(**"<link rel='stylesheet' type='text/css' href='/styles.css'>"**)** # html line: references the file used for CSS styling of the webpage (objective 4.2).

**print(**"<meta name='viewport' content='width=device-width, initial-scale=1'>"**)** # html line: formats the webpage so that the content atomatically adjusts its size to fit the screen.

**print** **(**"</head>"**)** # html line: the closing </head> element for structuring the web page.

**print(**"<body>"**)** # html line: the <body> element for structuring the web page.

# Objective 3.1:

**print(**"<div class='navbar'>"**)** # html line: references the navbar div defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<a href='/journeyplanner.html'class='left'>Journey Planner</a>"**)** # html line: the first part of the navbar is for the journey planner - when clicked, the website redirects to the html for the journey planner page.

**print(**"<a href='/map.html'class='center'>Map</a>"**)** # html line: the second part of the navbar is for the map - when clicked, the website redirects to the html for the map page.

**print(**"<a href='/updates.html'class='center'>Status Updates</a>"**)** # html line: the third part of the navbar is for status updates - when clicked, the website redirects to the html for the status updates page.

**print(**"<a href='/arrivaltimes.html'>Arrival Times</a>"**)** # html line: the fourth part of the navbar is for the arrival times - when clicked, the website redirects to the html for the arrival times page.

**print(**"<a href='/liftupdates.html'>Lift Updates</a>"**)** # html line: the fifth part of the navbar is for the lift updates - when clicked, the website redirects to the html for the lift updates page.

**print(**"<a href='/index.html'class='right'>About</a>"**)** # html line: the last part of the navbar is for the information page - when clicked, the website redirects to the html for the about page.

**print(**"</div>"**)** # html line: closing tag for the navbar div defined from the CSS file for structuring the web page.

**print(**"<div class='header'>"**)** # html line: references the header div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Tube Journey Planner</h3>"**)** # html line: text within the header div element is output.

**print(**"<h3>Farheen Fahim</h3>"**)** # html line: text within the header div element is output.

**print(**"</div>"**)** # html line: closes the header div element defined from the CSS file for structuring the web page.

**print** **(**"<h1>Route</h1>"**)** # html line: outputs a title for the webpage.

**class** **JourneyPlanner:** # Creates a class for the journey planner called "JourneyPlanner".

**def** \_\_init\_\_ **(**self**,**station1**,**station2**,**rush\_hour**,**weekend**):** # Defines the constructor method which takes 4 arguments (other than the object parameter).

# This method is used to define class attributes.

self**.**station1 **=** station1 # Assigns value stored in the parameter "station1" as the "self.station1" attribute.

self**.**station2 **=** station2 # Assigns value stored in the parameter "station2" as the "self.station2" attribute.

self**.**rush\_hour **=** rush\_hour # Assigns value stored in the parameter "rush\_hour" as the "self.rush\_hour" attribute.

self**.**weekend **=** weekend # Assigns value stored in the parameter "weekend" as the "self.weekend" attribute.

# Objectives 1.3, 1.4, 1.5:

# Objectives 2.1, 2.2:

# Objective 7:

# Creates an adjacency dictionary attribute called "self.stations\_adjacency\_dict" containing keys which represent the station you are at, and values which are dictionaries as well.

# These dictionaries contain keys which represent the station you can travel to, and the values are the number of seconds taken to travel there.

self**.**stations\_adjacency\_dict**=** **{**

"Paddington CircleLine"**:** **{**"Bayswater CircleLine"**:**85.7**,**"Edgware Road CircleLine"**:**110.8**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington DistrictLine"**:** 300**,**"Paddington HammersmithLine"**:**300**},**

"Paddington DistrictLine"**:** **{**"Bayswater DistrictLine"**:**85.7**,**"Edgware Road DistrictLine"**:**110.8**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington CircleLine"**:** 300**,**"Paddington HammersmithLine"**:**300**},**

"Paddington HammersmithLine"**:{**"Edgware Road HammersmithLine"**:**110.8**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington CircleLine"**:** 300**,**"Paddington DistrictLine"**:**300**},**

"Paddington BakerlooLine"**:** **{**"Edgware Road BakerlooLine"**:**75.3**,** "Paddington ElizabethLine"**:** 300**,**"Paddington DistrictLine"**:** 300**,**"Paddington CircleLine"**:** 300**,**"Paddington HammersmithLine"**:**300**},**

"Paddington ElizabethLine"**:** **{**"Tottenham Court Road ElizabethLine"**:**241.6**,**"Paddington BakerlooLine"**:**300**,**"Paddington DistrictLine"**:** 300**,**"Paddington CircleLine"**:** 300**,**"Paddington HammersmithLine"**:**300**},**

"Bayswater CircleLine"**:** **{**"Paddington CircleLine"**:**85.7**,** "Notting Hill Gate CircleLine"**:**86.1**,**"Bayswater DistrictLine"**:**300**},**

"Bayswater DistrictLine"**:** **{**"Paddington DistrictLine"**:**85.7**,** "Notting Hill Gate DistrictLine"**:**86.1**,**"Bayswater CircleLine"**:**300**},**

"Notting Hill Gate CircleLine"**:** **{**"Bayswater CircleLine"**:**86.1**,** "High Street Kensington CircleLine"**:**87.3**,** "Notting Hill Gate CentralLine"**:**300**,**"Notting Hill Gate DistrictLine"**:**300**},**

"Notting Hill Gate DistrictLine"**:** **{**"Bayswater DistrictLine"**:**86.1**,** "High Street Kensington DistrictLine"**:**87.3**,** "Notting Hill Gate CentralLine"**:**300**,**"Notting Hill Gate CircleLine"**:**300**},**

"Notting Hill Gate CentralLine"**:{**"Queensway CentralLine"**:**68.5**,** "Notting Hill Gate CircleLine"**:**300**,**"Notting Hill Gate DistrictLine"**:**300**},**

"High Street Kensington CircleLine"**:** **{**"Gloucester Road CircleLine"**:**150.1**,**"Notting Hill Gate CircleLine"**:**87.3**,**"High Street Kensington DistrictLine"**:**300**},**

"High Street Kensington DistrictLine"**:** **{**"Earl's Court DistrictLine"**:**175.6**,**"Notting Hill Gate DistrictLine"**:** 87.3**,**"High Street Kensington CircleLine"**:** 300**},**

"Earl's Court DistrictLine"**:** **{**"High Street Kensington DistrictLine"**:**175.6**,** "Gloucester Road DistrictLine"**:**91.4**,** "Earl's Court PiccadillyLine"**:**300**},**

"Earl's Court PiccadillyLine"**:** **{**"Gloucester Road PiccadillyLine"**:**91.4**,** "Earl's Court DistrictLine"**:**300**},**

"Gloucester Road PiccadillyLine"**:** **{**"Earl's Court PiccadillyLine"**:**91.4**,**"South Kensington PiccadillyLine"**:**76.6**,**"Gloucester Road CircleLine"**:**300**,**"Gloucester Road DistrictLine"**:**300**},**

"Gloucester Road CircleLine"**:{**"South Kensington CircleLine"**:**78.2**,** "High Street Kensington CircleLine"**:**150.1**,**"Gloucester Road DistrictLine"**:**300**,**"Gloucester Road PiccadillyLine"**:**300**},**

"Gloucester Road DistrictLine"**:{**"South Kensington DistrictLine"**:**78.2**,** "Earl's Court DistrictLine"**:**91.4**,**"Gloucester Road CircleLine"**:**300**,**"Gloucester Road PiccadillyLine"**:**300**},**

"South Kensington CircleLine"**:** **{**"Gloucester Road CircleLine"**:**78.2**,**"Sloane Square CircleLine"**:**110.4**,**"South Kensington DistrictLine"**:**300**,**"South Kensington PiccadillyLine"**:**300**},**

"South Kensington DistrictLine"**:{**"Gloucester Road DistrictLine"**:**78.2**,** "Sloane Square DistrictLine"**:**110.4 **,** "South Kensington CircleLine"**:**300**,**"South Kensington PiccadillyLine"**:**300**},**

"South Kensington PiccadillyLine"**:** **{**"Knightsbridge PiccadillyLine"**:**156.8**,**"Gloucester Road PiccadillyLine"**:**76.6**,**"South Kensington DistrictLine"**:**300**,**"South Kensington CircleLine"**:**300**},**

"Sloane Square CircleLine"**:** **{**"South Kensington CircleLine"**:**110.4**,**"VictoriaStation CircleLine"**:**88.7**,**"Sloane Square DistrictLine"**:**300**},**

"Sloane Square DistrictLine"**:** **{**"South Kensington DistrictLine"**:**110.4**,**"VictoriaStation DistrictLine"**:**88.7**,**"Sloane Square CircleLine"**:**300**},**

"VictoriaStation CircleLine"**:** **{**"Sloane Square CircleLine"**:**88.7**,**"St. James's Park CircleLine"**:**75.1**,**"VictoriaStation DistrictLine"**:**300**,**"VictoriaStation VictoriaLine"**:**300**},**

"VictoriaStation DistrictLine"**:** **{**"Sloane Square DistrictLine"**:**88.7**,**"St. James's Park DistrictLine"**:**75.1**,**"VictoriaStation CircleLine"**:**300**,**"VictoriaStation VictoriaLine"**:**300**},**

"VictoriaStation VictoriaLine"**:** **{**"Pimlico VictoriaLine"**:**82.3**,**"Green Park VictoriaLine"**:**78.4**,**"VictoriaStation CircleLine"**:**300**,**"VictoriaStation DistrictLine"**:**300**},**

"St. James's Park CircleLine"**:** **{**"VictoriaStation CircleLine"**:**75.1**,**"Westminster CircleLine"**:**86.1**,**"St. James's Park DistrictLine"**:**300**},**

"St. James's Park DistrictLine"**:** **{**"VictoriaStation DistrictLine"**:**75.1**,**"Westminster DistrictLine"**:**86.1**,**"St. James's Park CircleLine"**:**300**},**

"Westminster CircleLine"**:** **{**"St. James's Park CircleLine"**:**86.1**,**"Embankment CircleLine"**:**81.6**,**"Westminster DistrictLine"**:**300**,**"Westminster JubileeLine"**:**300**},**

"Westminster DistrictLine"**:** **{**"St. James's Park DistrictLine"**:**86.1**,**"Embankment DistrictLine"**:**81.6**,**"Westminster CircleLine"**:**300**,**"Westminster JubileeLine"**:**300**},**

"Westminster JubileeLine"**:** **{**"WaterlooStation JubileeLine"**:**78.1**,**"Green Park JubileeLine"**:**99.8**,**"Westminster DistrictLine"**:**300**,**"Westminster CircleLine"**:**300**},**

"Embankment CircleLine"**:** **{**"Westminster CircleLine"**:**81.6**,**"Temple CircleLine"**:**66.4**,**"Embankment DistrictLine"**:**300**,**"Embankment BakerlooLine"**:**300**,**"Embankment NorthernLine"**:**300**},**

"Embankment DistrictLine"**:** **{**"Westminster DistrictLine"**:**81.6**,**"Temple DistrictLine"**:**66.4**,**"Embankment CircleLine"**:**300**,**"Embankment BakerlooLine"**:**300**,**"Embankment NorthernLine"**:**300**},**

"Embankment BakerlooLine"**:** **{**"WaterlooStation BakerlooLine"**:**49.5**,**"Charing Cross BakerlooLine"**:**93.4**,**"Embankment DistrictLine"**:**300**,**"Embankment CircleLine"**:**300**,**"Embankment NorthernLine"**:**300**},**

"Embankment NorthernLine"**:** **{**"WaterlooStation NorthernLine"**:**49.5**,**"Charing Cross NorthernLine"**:**93.4**,**"Embankment DistrictLine"**:**300**,**"Embankment BakerlooLine"**:**300**,**"Embankment CircleLine"**:**300**},**

"Temple CircleLine"**:** **{**"Embankment CircleLine"**:**66.4**,**"Blackfriars CircleLine"**:**86.6**,**"Temple DistrictLine"**:**300**},**

"Temple DistrictLine"**:** **{**"Embankment DistrictLine"**:**66.4**,**"Blackfriars DistrictLine"**:**86.6**,**"Temple CircleLine"**:**300**},**

"Blackfriars CircleLine"**:** **{**"Temple CircleLine"**:**86.6**,**"Mansion House CircleLine"**:**90.9**,**"Blackfriars DistrictLine"**:**300**},**

"Blackfriars DistrictLine"**:** **{**"Temple DistrictLine"**:**86.6**,**"Mansion House DistrictLine"**:**90.9**,**"Blackfriars CircleLine"**:**300**},**

"Mansion House CircleLine"**:** **{**"Blackfriars CircleLine"**:**90.9**,**"Cannon Street CircleLine"**:**61.9**,**"Mansion House DistrictLine"**:**300**},**

"Mansion House DistrictLine"**:** **{**"Blackfriars DistrictLine"**:**90.9**,**"Cannon Street DistrictLine"**:**61.9**,**"Mansion House CircleLine"**:**300**},**

"Cannon Street CircleLine"**:** **{**"Mansion House CircleLine"**:**61.9**,**"Monument CircleLine"**:**66.3**,**"Cannon Street DistrictLine"**:**300**},**

"Cannon Street DistrictLine"**:** **{**"Mansion House DistrictLine"**:**61.9**,**"Monument DistrictLine"**:**66.3**,**"Cannon Street CircleLine"**:**300**},**

"Monument CircleLine"**:** **{**"Cannon Street CircleLine"**:**66.3**,**"Tower Hill CircleLine"**:**92.5**,**"Monument DistrictLine"**:**300**},**

"Monument DistrictLine"**:** **{**"Cannon Street DistrictLine"**:**66.3**,**"Tower Hill DistrictLine"**:**92.5**,**"Monument CircleLine"**:**300**},**

"Tower Hill CircleLine"**:** **{**"Monument CircleLine"**:**92.5**,**"AldgateStation CircleLine"**:**95.9**,**"Tower Hill DistrictLine"**:**300**},**

"Tower Hill DistrictLine"**:** **{**"Monument DistrictLine"**:**92.5**,**"Aldgate East DistrictLine"**:**132.9**,**"Tower Hill CircleLine"**:**300**},**

"Aldgate East DistrictLine"**:** **{**"Tower Hill DistrictLine"**:**132.9**,**"Aldgate East HammersmithLine"**:**300**},**

"Aldgate East HammersmithLine"**:{**"Liverpool Street HammersmithLine"**:**128.4**,**"Aldgate East DistrictLine"**:**300**},**

"AldgateStation CircleLine"**:** **{**"Tower Hill CircleLine"**:**95.9**,**"Liverpool Street CircleLine"**:**102.9**},**

"Liverpool Street CircleLine"**:** **{**"AldgateStation CircleLine"**:**102.9**,**"Moorgate CircleLine"**:**70.6**,**"Liverpool Street HammersmithLine"**:**300**,**"Liverpool Street CentralLine"**:**300**,**"Liverpool Street ElizabethLine"**:**300**},**

"Liverpool Street HammersmithLine"**:** **{**"Aldgate East HammersmithLine"**:**128.4**,**"Moorgate HammersmithLine"**:**70.6**,**"Liverpool Street CircleLine"**:**300**,**"Liverpool Street CentralLine"**:**300**,**"Liverpool Street ElizabethLine"**:**300**},**

"Liverpool Street CentralLine"**:{**"Bank CentralLine"**:**97.5**,**"Liverpool Street CircleLine"**:**300**,**"Liverpool Street HammersmithLine"**:**300**,**"Liverpool Street ElizabethLine"**:**300**},**

"Liverpool Street ElizabethLine"**:{**"Farringdon ElizabethLine"**:**96.0**,** "Liverpool Street CircleLine"**:**300**,**"Liverpool Street HammersmithLine"**:**300**,**"Liverpool Street CentralLine"**:**300**},**

"Moorgate CircleLine"**:** **{**"Liverpool Street CircleLine"**:**70.6**,**"Barbican CircleLine"**:**72.1**,**"Moorgate HammersmithLine"**:**300**,**"Moorgate NorthernLine"**:**300**},**

"Moorgate HammersmithLine"**:** **{**"Liverpool Street HammersmithLine"**:**70.6**,**"Barbican HammersmithLine"**:**72.1**,**"Moorgate CircleLine"**:**300**,**"Moorgate NorthernLine"**:**300**},**

"Moorgate NorthernLine"**:** **{**"Bank NorthernLine"**:**88.5**,**"Old Street NorthernLine"**:**99.5**,**"Moorgate CircleLine"**:**300**,**"Moorgate HammersmithLine"**:**300**},**

"Barbican CircleLine"**:** **{**"Moorgate CircleLine"**:**72.1**,**"Farringdon CircleLine"**:**67.6**,**"Barbican HammersmithLine"**:**300**},**

"Barbican HammersmithLine"**:** **{**"Moorgate HammersmithLine"**:**72.1**,**"Farringdon HammersmithLine"**:**67.6**,**"Barbican CircleLine"**:**300**},**

"Farringdon ElizabethLine"**:** **{**"Liverpool Street ElizabethLine"**:**96.0**,**"Tottenham Court Road ElizabethLine"**:**138.8**,**"Farringdon CircleLine"**:**300**,**"Farringdon HammersmithLine"**:**300**},**

"Farringdon CircleLine"**:** **{**"Barbican CircleLine"**:**67.6**,**"King's Cross St. Pancras CircleLine"**:**164.0**,**"Farringdon ElizabethLine"**:**300**,**"Farringdon HammersmithLine"**:**300**},**

"Farringdon HammersmithLine"**:** **{**"Barbican HammersmithLine"**:**67.6**,**"King's Cross St. Pancras HammersmithLine"**:**164.0**,**"Farringdon ElizabethLine"**:**300**,**"Farringdon CircleLine"**:**300**},**

"King's Cross St. Pancras CircleLine"**:** **{**"Farringdon CircleLine"**:**164.0**,**"Euston Square CircleLine"**:**83.2**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**},**

"King's Cross St. Pancras HammersmithLine"**:** **{**"Farringdon HammersmithLine"**:**164.0**,**"Euston Square HammersmithLine"**:**83.2**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**},**

"King's Cross St. Pancras NorthernLine"**:** **{**"EustonStation NorthernLine"**:**95.8**,**"Angel NorthernLine"**:**119.7**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**},**

"King's Cross St. Pancras VictoriaLine"**:** **{**"EustonStation VictoriaLine"**:**87.4**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**},**

"King's Cross St. Pancras PiccadillyLine"**:** **{**"Russell Square PiccadillyLine"**:**89.2**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras CircleLine"**:**300**},**

"Euston Square CircleLine"**:** **{**"King's Cross St. Pancras CircleLine"**:**83.2**,**"Great Portland Street CircleLine"**:**65.9**,**"Euston Square HammersmithLine"**:**300**},**

"Euston Square HammersmithLine"**:** **{**"King's Cross St. Pancras HammersmithLine"**:**83.2**,**"Great Portland Street HammersmithLine"**:**65.9**,**"Euston Square CircleLine"**:**300**},**

"Great Portland Street CircleLine"**:** **{**"Euston Square CircleLine"**:**65.9**,**"Baker Street CircleLine"**:**88.4**,**"Great Portland Street HammersmithLine"**:**300**},**

"Great Portland Street HammersmithLine"**:** **{**"Euston Square HammersmithLine"**:**65.9**,**"Baker Street HammersmithLine"**:**88.4**,**"Great Portland Street CircleLine"**:**300**},**

"Baker Street CircleLine"**:** **{**"Great Portland Street CircleLine"**:**88.4**,**"Edgware Road CircleLine"**:**110.7**,**"Baker Street HammersmithLine"**:**300**,**"Baker Street BakerlooLine"**:**300**,**"Baker Street JubileeLine"**:**300**},**

"Baker Street HammersmithLine"**:** **{**"Great Portland Street HammersmithLine"**:**88.4**,**"Edgware Road HammersmithLine"**:**110.7**,**"Baker Street CircleLine"**:**300**,**"Baker Street BakerlooLine"**:**300**,**"Baker Street JubileeLine"**:**300**},**

"Baker Street BakerlooLine"**:** **{**"Marylebone BakerlooLine"**:**64.3**,**"Regent's Park BakerlooLine"**:**98.9**,**"Baker Street HammersmithLine"**:**300**,**"Baker Street CircleLine"**:**300**,**"Baker Street JubileeLine"**:**300**},**

"Baker Street JubileeLine"**:** **{**"Bond Street JubileeLine"**:**112.5**,**"Baker Street HammersmithLine"**:**300**,**"Baker Street BakerlooLine"**:**300**,**"Baker Street CircleLine"**:**300**},**

"Edgware Road CircleLine"**:** **{**"Baker Street CircleLine"**:**110.7**,**"Paddington CircleLine"**:**110.8**,**"Edgware Road HammersmithLine"**:**300**,**"Edgware Road DistrictLine"**:**300**,**"Edgware Road BakerlooLine"**:**300**},**

"Edgware Road HammersmithLine"**:** **{**"Baker Street HammersmithLine"**:**110.7**,**"Paddington HammersmithLine"**:**110.8**,**"Edgware Road CircleLine"**:**300**,**"Edgware Road DistrictLine"**:**300**,**"Edgware Road BakerlooLine"**:**300**},**

"Edgware Road DistrictLine"**:** **{**"Paddington DistrictLine"**:**93.1**,**"Edgware Road HammersmithLine"**:**300**,**"Edgware Road CircleLine"**:**300**,**"Edgware Road BakerlooLine"**:**300**},**

"Edgware Road BakerlooLine"**:** **{**"Paddington BakerlooLine"**:**75.3**,**"Marylebone BakerlooLine"**:**69.1**,**"Edgware Road HammersmithLine"**:**300**,**"Edgware Road DistrictLine"**:**300**,**"Edgware Road CircleLine"**:**300**},**

"Marylebone BakerlooLine"**:** **{**"Edgware Road BakerlooLine"**:**69.1**,**"Baker Street BakerlooLine"**:**64.3**},**

"Angel NorthernLine"**:** **{**"King's Cross St. Pancras NorthernLine"**:**119.7**,**"Old Street NorthernLine"**:**122.4**},**

"Old Street NorthernLine"**:** **{**"Angel NorthernLine"**:**122.4**,**"Moorgate NorthernLine"**:**99.5**},**

"EustonStation NorthernLine"**:** **{**"King's Cross St. Pancras NorthernLine"**:**95.8**,**"Warren Street NorthernLine"**:**65.6**,**"EustonStation VictoriaLine"**:**300**},**

"EustonStation VictoriaLine"**:** **{**"King's Cross St. Pancras VictoriaLine"**:**87.4**,**"Warren Street VictoriaLine"**:**58.7**,**"EustonStation NorthernLine"**:**300**},**

"Queensway CentralLine"**:** **{**"Notting Hill Gate CentralLine"**:**68.5**,**"Lancaster Gate CentralLine"**:**98.6**},**

"Lancaster Gate CentralLine"**:** **{**"Queensway CentralLine"**:**98.6**,**"Marble Arch CentralLine"**:**101.2**},**

"Marble Arch CentralLine"**:** **{**"Lancaster Gate CentralLine"**:**101.2**,**"Bond Street CentralLine"**:**100.5**},**

"Bond Street CentralLine"**:** **{**"Marble Arch CentralLine"**:**100.5**,**"Oxford Circus CentralLine"**:**83.3**,**"Bond Street JubileeLine"**:**300**},**

"Bond Street JubileeLine"**:** **{**"Baker Street JubileeLine"**:**112.5**,**"Green Park JubileeLine"**:**94.3**,**"Bond Street CentralLine"**:**300**},**

"Green Park JubileeLine"**:** **{**"Bond Street JubileeLine"**:**94.3**,**"Westminster JubileeLine"**:**99.8**,**"Green Park PiccadillyLine"**:**300**,**"Green Park VictoriaLine"**:**300**},**

"Green Park PiccadillyLine"**:** **{**"Hyde Park Corner PiccadillyLine"**:**116.1**,**"PiccadillyCircus PiccadillyLine"**:**67.0**,**"Green Park JubileeLine"**:**300**,**"Green Park VictoriaLine"**:**300**},**

"Green Park VictoriaLine"**:** **{**"Oxford Circus VictoriaLine"**:**77.6**,**"VictoriaStation VictoriaLine"**:**78.4**,**"Green Park PiccadillyLine"**:**300**,**"Green Park JubileeLine"**:**300**},**

"Knightsbridge PiccadillyLine"**:** **{**"South Kensington PiccadillyLine"**:**156.8**,**"Hyde Park Corner PiccadillyLine"**:**71.0**},**

"Hyde Park Corner PiccadillyLine"**:** **{**"Knightsbridge PiccadillyLine"**:**71.0**,**"Green Park PiccadillyLine"**:**116.1**},**

"PiccadillyCircus PiccadillyLine"**:** **{**"Green Park PiccadillyLine"**:**67.0**,**"Leicester Square PiccadillyLine"**:**68.9**,**"PiccadillyCircus BakerlooLine"**:**300**},**

"PiccadillyCircus BakerlooLine"**:** **{**"Oxford Circus BakerlooLine"**:**99.5**,**"Charing Cross BakerlooLine"**:**66.5**,**"PiccadillyCircus PiccadillyLine"**:**300**},**

"Oxford Circus CentralLine"**:** **{**"Bond Street CentralLine"**:**83.3**,**"Tottenham Court Road CentralLine"**:**70.3**,**"Oxford Circus BakerlooLine"**:**300**,**"Oxford Circus VictoriaLine"**:**300**},**

"Oxford Circus BakerlooLine"**:** **{**"PiccadillyCircus BakerlooLine"**:**99.5**,**"Regent's Park BakerlooLine"**:**105.8**,**"Oxford Circus CentralLine"**:**300**,**"Oxford Circus VictoriaLine"**:**300**},**

"Oxford Circus VictoriaLine"**:** **{**"Green Park VictoriaLine"**:**77.6**,**"Warren Street VictoriaLine"**:**71.5**,**"Oxford Circus BakerlooLine"**:**300**,**"Oxford Circus CentralLine"**:**300**},**

"Regent's Park BakerlooLine"**:** **{**"Baker Street BakerlooLine"**:**98.9**,**"Oxford Circus BakerlooLine"**:**105.8**},**

"Tottenham Court Road CentralLine"**:** **{**"Oxford Circus CentralLine"**:**70.3**,**"Holborn CentralLine"**:**79.9**,**"Tottenham Court Road ElizabethLine"**:**300**,**"Tottenham Court Road NorthernLine"**:**300**},**

"Tottenham Court Road ElizabethLine"**:** **{**"Farringdon ElizabethLine"**:**138.8**,**"Paddington ElizabethLine"**:**241.6**,**"Tottenham Court Road CentralLine"**:**300**,**"Tottenham Court Road NorthernLine"**:**300**},**

"Tottenham Court Road NorthernLine"**:** **{**"Goodge Street NorthernLine"**:**63.2**,**"Leicester Square NorthernLine"**:**49.6**,**"Tottenham Court Road ElizabethLine"**:**300**,**"Tottenham Court Road CentralLine"**:**300**},**

"Goodge Street NorthernLine"**:** **{**"Tottenham Court Road NorthernLine"**:**63.2**,**"Warren Street NorthernLine"**:**50.9**},**

"Warren Street NorthernLine"**:** **{**"Goodge Street NorthernLine"**:**50.9**,**"EustonStation NorthernLine"**:**65.6**,**"Warren Street VictoriaLine"**:**300**},**

"Warren Street VictoriaLine"**:** **{**"EustonStation VictoriaLine"**:**58.7**,**"Oxford Circus VictoriaLine"**:**71.5**,**"Warren Street NorthernLine"**:**300**},**

"Holborn CentralLine"**:** **{**"Tottenham Court Road CentralLine"**:**79.9**,**"Chancery Lane CentralLine"**:**48.5**,**"Holborn PiccadillyLine"**:**300**},**

"Holborn PiccadillyLine"**:** **{**"Russell Square PiccadillyLine"**:**84.7**,**"Covent Garden PiccadillyLine"**:**86.7**,**"Holborn CentralLine"**:**300**},**

"Russell Square PiccadillyLine"**:** **{**"King's Cross St. Pancras PiccadillyLine"**:**89.2**,**"Holborn PiccadillyLine"**:**84.7**},**

"Leicester Square NorthernLine"**:** **{**"Tottenham Court Road NorthernLine"**:**49.6**,**"Charing Cross NorthernLine"**:**67.6**,**"Leicester Square PiccadillyLine"**:**300**},**

"Leicester Square PiccadillyLine"**:** **{**"PiccadillyCircus PiccadillyLine"**:**68.9**,**"Covent Garden PiccadillyLine"**:**44.9**,**"Leicester Square NorthernLine"**:**300**},**

"Covent Garden PiccadillyLine"**:** **{**"Leicester Square PiccadillyLine"**:**44.9**,**"Holborn PiccadillyLine"**:**86.7**},**

"Chancery Lane CentralLine"**:** **{**"Holborn CentralLine"**:**48.5**,**"St. Paul's CentralLine"**:**87.0**},**

"Charing Cross NorthernLine"**:** **{**"Embankment NorthernLine"**:**93.4**,**"Leicester Square NorthernLine"**:**67.6**,**"Charing Cross BakerlooLine"**:**300**},**

"Charing Cross BakerlooLine"**:** **{**"Embankment BakerlooLine"**:**50.4**,**"PiccadillyCircus BakerlooLine"**:**66.5**,**"Charing Cross NorthernLine"**:**300**},**

"WaterlooStation JubileeLine"**:** **{**"Westminster JubileeLine"**:**78.1**,**"Southwark JubileeLine"**:**49.5**,**"WaterlooStation NorthernLine"**:**300**,**"WaterlooStation BakerlooLine"**:**300**,**"WaterlooStation WaterlooLine"**:**300**},**

"WaterlooStation NorthernLine"**:** **{**"Kennington NorthernLine"**:**169.8**,**"Embankment NorthernLine"**:**49.5**,**"WaterlooStation JubileeLine"**:**300**,**"WaterlooStation BakerlooLine"**:**300**,**"WaterlooStation WaterlooLine"**:**300**},**

"WaterlooStation BakerlooLine"**:** **{**"Lambeth North BakerlooLine"**:**87.8**,**"Embankment BakerlooLine"**:**50.4**,**"WaterlooStation NorthernLine"**:**300**,**"WaterlooStation JubileeLine"**:**300**,**"WaterlooStation WaterlooLine"**:**300**},**

"WaterlooStation WaterlooLine"**:** **{**"Bank WaterlooLine"**:**213.4**,**"WaterlooStation NorthernLine"**:**300**,**"WaterlooStation BakerlooLine"**:**300**,**"WaterlooStation JubileeLine"**:**300**},**

"Southwark JubileeLine"**:** **{**"WaterlooStation JubileeLine"**:**49.5**,**"London Bridge JubileeLine"**:**92.2**},**

"London Bridge JubileeLine"**:** **{**"Southwark JubileeLine"**:**92.2**,**"London Bridge NorthernLine"**:**300**},**

"London Bridge NorthernLine"**:** **{**"Bank NorthernLine"**:**79.8**,**"Borough NorthernLine"**:**79.0**,**"London Bridge JubileeLine"**:**300**},**

"Borough NorthernLine"**:** **{**"Elephant & Castle NorthernLine"**:**94.2**,**"London Bridge NorthernLine"**:**79.0**},**

"Lambeth North BakerlooLine"**:** **{**"WaterlooStation BakerlooLine"**:**87.8**,**"Elephant & Castle BakerlooLine"**:**151.1**},**

"Elephant & Castle NorthernLine"**:** **{**"Borough NorthernLine"**:**94.2**,**"Kennington NorthernLine"**:**100.7**,**"Elephant & Castle BakerlooLine"**:**300**},**

"Elephant & Castle BakerlooLine"**:** **{**"Lambeth North BakerlooLine"**:**151.1**,**"Elephant & Castle NorthernLine"**:**300**},**

"St. Paul's CentralLine"**:** **{**"Chancery Lane CentralLine"**:**87.0**,**"Bank CentralLine"**:**96.3**},**

"Bank CentralLine"**:** **{**"St. Paul's CentralLine"**:**96.3**,**"Liverpool Street CentralLine"**:**97.5**,**"Bank NorthernLine"**:**300**,**"Bank WaterlooLine"**:**300**},**

"Bank NorthernLine"**:** **{**"London Bridge NorthernLine"**:**79.8**,**"Moorgate NorthernLine"**:**88.5**,**"Bank CentralLine"**:**300**,**"Bank WaterlooLine"**:**300**},**

"Bank WaterlooLine"**:** **{**"WaterlooStation WaterlooLine"**:**213.4**,**"Bank NorthernLine"**:**300**,**"Bank CentralLine"**:**300**},**

"Battersea Power Station NorthernLine"**:** **{**"Nine Elms NorthernLine"**:**98.8**},**

"Nine Elms NorthernLine"**:** **{**"Battersea Power Station NorthernLine"**:**98.8**,**"Kennington NorthernLine"**:**168.5**},**

"Kennington NorthernLine"**:** **{**"Nine Elms NorthernLine"**:**168.5**,**"WaterlooStation NorthernLine"**:**169.8**,**"Elephant & Castle NorthernLine"**:**100.7**},**

"Pimlico VictoriaLine"**:** **{**"Vauxhall VictoriaLine"**:**62.1**,**"VictoriaStation VictoriaLine"**:**82.3**},**

"Vauxhall VictoriaLine"**:** **{**"Pimlico VictoriaLine"**:**62.1**}**

**}**

self**.**stations\_list **=** **list(**self**.**stations\_adjacency\_dict**.**keys**())** # Assigns keys of the stations adjacency dictionary as the "self.stations\_list" attribute.

self**.**num\_of\_stations **=** **len(**self**.**stations\_list**)** # Assigns the length of the stations adjacency dictionary as the "self.num\_of\_stations" attribute.

self**.**stepfree **=** **False** # Assigns the Boolean False value as the "self.stepfree" attribute.

**def** correct\_stations**(**self**):** # Defines the "correct\_stations" method which takes no arguments (other than the object parameter).

# This method is used to adjust any stations entered as they may include the line of the station or other station names within it. This will not work when inputted in the "dijkstra" function because the function searches for specific phrases within strings.

# For example, if the user enters "Victoria" station as their start station, this must be chnaged to "VictoriaStation" in order to ensure that it is not confused with the Victoria line.

# This method uses if-elif statements to change values which the user may have inputted so that they are compatible with the "dijkstra" function.

**if** self**.**station1 **==** "Victoria"**:**

self**.**station1 **=** "VictoriaStation"

**elif** self**.**station1 **==** "Piccadilly Circus"**:**

self**.**station1 **=** "PiccadillyCircus"

**elif** self**.**station1 **==** "Waterloo"**:**

self**.**station1 **=** "WaterlooStation"

**elif** self**.**station1 **==** "Aldgate"**:**

self**.**station1 **=** "AldgateStation"

**elif** self**.**station1 **==** "Euston"**:**

self**.**station1 **=** "EustonStation"

**if** self**.**station2 **==** "Victoria"**:**

self**.**station2 **=** "VictoriaStation"

**elif** self**.**station2 **==** "Piccadilly Circus"**:**

self**.**station2 **=** "PiccadillyCircus"

**elif** self**.**station2 **==** "Waterloo"**:**

self**.**station2 **=** "WaterlooStation"

**elif** self**.**station1 **==** "Aldgate"**:**

self**.**station1 **=** "AldgateStation"

**elif** self**.**station2 **==** "Aldgate"**:**

self**.**station2 **=** "AldgateStation"

**elif** self**.**station2 **==** "Euston"**:**

self**.**station2 **=** "EustonStation"

# Objective 2:

**def** dijkstra**(**self**,**start\_node**,**end\_node**):** # Defines the "dijkstra" method which takes 2 arguments (other than the object parameter).

# This method is used to calculate the shortest route between the parameter called "start\_node" and the parameter called "end\_node".

# It determines the index position of "start\_node" in "self.stations\_list" using the "linear\_search" method.

# Then, empty dictionaries are created which correspond to the nodes that have been visited ("visited\_nodes"), the station used to reach another station ("path\_taken"), and the distances between the "start\_node" and every other node ("distances").

# All of the nodes are marked as unvisited in the "visited\_nodes" dictionary by making the keys the elements in the list called "self.stations\_list" and the values corresponding to them as False.

# All of the distances to each node are set to infinity in the "distances" dictionary by making the keys the elements in the list called "self.stations\_list" and the values corresponding to them as math.inf.

# The distance to the start node is set to 0 by making the value corresponding to the "start\_node" key as 0 in the "distances" dictionary.

# Based on whether it is rush hour or not and if it is the weekned, a variable called multiplier is determined and then all of the values in the adjacency dictionary are multiplied by that number.

# The next part of code is run fro every node/station, so it uses a for loop.

# The closest unvisited node is selected by using a for loop (nested within another for loop) and comparing values in the "distances" dictionary and checking if the corresponding values in the "visited\_nodes" dictionary are False or not.

# After the closest unvisited node is selected, it is marked as visited by changing its corresponding value in the "visited\_nodes" dictionary.

# Then, the distances to the nodes adjacent to the selected node are calculated. The distance is calculated by adding the distance between the selected node and the adjacent node and the distance between the start node and the selected node.

# The distances in the dictionary called "distances" are updated if the value caulculated is less than the value already stored.

# If the value is updates, then a record of how to get to that node is made using the dictionary called "path\_taken".

# After the for loop has run and all distances to all other nodes are caulculated:

# If the journey is not step free, then the "find\_end\_node" function is called which is used to determine which line the user will use last. This ensures that the route does not have an added 300 seconds due to a changeover at the end station which is not required.

# A "path\_list" is created which contains elements which are part of the route the user should take.

# The list called "path\_list" is the returned value from the "recursive\_find\_path" function.

# Then, "path\_list" is reversed (as it contains the route from the end node to the start node).

# The "distances" dictionary, "path\_list" list, and "end\_node" variable are returned from this function.

start\_node\_index **=** self**.**linear\_search**(**start\_node**,** self**.**stations\_list**)** # Calls the "linear\_Search" function with the parameters "start\_node" and "self.stations\_list", then stores the returned value in "start\_node\_index".

visited\_nodes **=** **{}** # Creates an empty dictionary called "visited\_nodes".

path\_taken **=** **{}** # Creates an empty dictionary called "path\_taken".

distances **=** **{}** # Creates an empty dictionary called "distances".

**for** i **in** **range** **(**self**.**num\_of\_stations**):** # For loop which iterates through "self.stations\_list".

visited\_nodes**[**self**.**stations\_list**[**i**]]** **=** **False** # Assigns the value corresponding to the key "self.stations\_list[i]" in the dictionary "visited\_nodes" to be False.

distances**[**self**.**stations\_list**[**i**]]** **=** math**.**inf # Assigns the value corresponding to the key "self.stations\_list[i]" in the dictionary "distances" to be infinity.

distances**[**start\_node**]** **=** 0 # Changes the value corresponding to the key "start\_node" in the dictionary "distances" to be equal to 0 as this dictionary contains the distances to get to that node from the start node.

# Objectives 5.1, 5.2, 5.3:

#Changes all times depending on if it is rush hour or a weekend:

**if** self**.**rush\_hour **==** **True** **and** self**.**weekend **==** **True:** # If "self.rush\_hour" is True and "self.weekend" is also True, then run the following lines of code.

multiplier **=** 2.5 # Assings the value of 2.5 to the variable called "multiplier".

**elif** self**.**rush\_hour **==** **False** **and** self**.**weekend **==** **True:** # If "self.rush\_hour" is False and "self.weekend" is True, then run the following lines of code.

multiplier **=** 1.5 # Assings the value of 1.5 to the variable called "multiplier".

**elif** self**.**rush\_hour **==** **True** **and** self**.**weekend **==** **False:** # If "self.rush\_hour" is True and "self.weekend" is False, then run the following lines of code.

multiplier **=** 2 # Assigns the value of 2 to the variable called "multiplier".

**else:** # If "self.rush\_hour" is False and "self.weekend" is also False, then run the following lines of code.

multiplier **=** 1 # Assigns the value of 1 to the variable called "multiplier".

**for** i **in** **range(**self**.**num\_of\_stations**):** # For loop which iterates through the "self.num\_of\_stations" list.

key **=** self**.**stations\_list**[**i**]** # Assigns the value in "self.stations\_list[i]" in the variable called "key".

**dict** **=** self**.**stations\_adjacency\_dict**[**key**]** # Stores the value corresponding to the key "key" in the dictionary called "self.stations\_adjacency\_dict" using the identifier called "dict".

**for** j **in** **range(**self**.**num\_of\_stations**):** # For loop which iterates through the "self.num\_of\_stations" list.

**try:** # Try block for error handling.

node\_key **=** self**.**stations\_list**[**j**]** # Assigns the value in "self.stations\_list[j]" in the variable called "node\_key".

**dict[**node\_key**]** **=** **dict[**node\_key**]** **\*** multiplier # Multiplies the value corresponding to the key called "node\_key" in the dictionary called "dict" by the variable called "multiplier".

**except:** # Except block for error handling.

**pass** # If there is an error, this line is run so that the error is handled.

**for** i **in** **range** **(**self**.**num\_of\_stations**):** # For loop which iterates through all of the stations in "self.stations\_list".

closest\_node\_index **=** **None** # Assigns None to the variable called "closest\_node\_index".

# SELECT CLOSEST UNVISITED NODE:

**for** j **in** **range(**self**.**num\_of\_stations**):** # For loop which iterates through all of the stations in "self.stations\_list" in order to determine which node is the closest and unvisited node.

**if** i **==** 0**:** # If i is equal to 0, then run the following code (as closest\_node\_index" will be None before this and you cannot use None for an index in a list).

closest\_node\_index **=** start\_node\_index # Assigns the value in "start\_node\_index" to the variable "closest\_node\_index".

closest\_node **=** self**.**stations\_list**[**closest\_node\_index**]** # Assigns "self.stations\_list[closest\_node\_index]" as the variable "closest\_node".

**elif** closest\_node\_index **==** **None:** # If closest\_node\_index is None, then run the following code (as you cannot use None for an index in a list).

**for** k **in** **range(**self**.**num\_of\_stations**):** # For loop which iterates through all of the stations in "self.stations\_list".

key **=** self**.**stations\_list**[**k**]** # Assigns "self.stations\_list[k]" to the variable "key".

**if** visited\_nodes**[**key**]** **==** **False** **and** distances**[**key**]** **!=** math**.**inf **and** distances**[**key**]** **!=** 0**:** # If the distance to the "key" is infinity but not 0, and the node is unvisited, run the following code.

closest\_node\_index **=** k # Assigns k to the variable "closest\_node\_index".

closest\_node **=** self**.**stations\_list**[**closest\_node\_index**]** # Assigns "self.stations\_list[closest\_node\_index]" to the variable "closest\_node".

**elif** distances**[**self**.**stations\_list**[**j**]]** **<** distances**[**closest\_node**]** **and** visited\_nodes**[**self**.**stations\_list**[**j**]]** **==** **False:** # If the current closest\_node has a distance greater than "distances[self.stations\_list[j]]", and this node is unvisited, run the following lines of code.

closest\_node\_index **=** j # Assigns j to the variable "closest\_node\_index".

closest\_node **=** self**.**stations\_list**[**closest\_node\_index**]** # Assigns "self.stations\_list[closest\_node\_index]" to the variable called "closest\_node".

current\_node **=** closest\_node # Assigns the value in "closest\_node" to the variable "current\_node".

current\_node\_index **=** closest\_node\_index # Assigns the value in "closest\_node\_index" to the variable "current\_node\_index".

# MARK NODE AS VISITED:

visited\_nodes**[**current\_node**]** **=** **True** # Changes the value corresponding to the key called "current\_node" in the dictionary called "visited\_nodes" to True.

adjacent\_nodes\_list **=** self**.**stations\_adjacency\_dict**[**current\_node**]** # Assigns the value corresponding to the key "current\_node" in the dictionary "self.stations\_adjacency\_dict" to the dictionary called "adjacent\_nodes\_list".

add\_on\_distance **=** distances**[**current\_node**]** # Stores the distance to the current\_node ("distances[current\_node]") in the variable called "add\_on\_distance".

# UPDATE VALUES IF THE DISTANCES ARE SHORTER:

keys\_list **=** **list(**adjacent\_nodes\_list**.**keys**())** # Stores the keys in the "adjacent\_nodes\_list" in "keys\_list" in a list format.

**for** key **in** **(**keys\_list**):** # For loop which iterates through every element in the list called "keys\_list".

new\_distance **=** adjacent\_nodes\_list**[**key**]** **+** add\_on\_distance # Adds the value stored in "adjacent\_nodes\_list[key]" to "add\_on\_distance" and stores the result to the variable called "new\_distance".

#Objective 1.2:

**if** new\_distance **<** distances**[**key**]** **and** visited\_nodes**[**key**]** **==** **False:** # If the value stored in "new\_distance" is less than the distance in "distances[key]" (the distance alreadfy recorded) and "visited\_nodes[key]" is False, then run the following code.

distances**[**key**]** **=** new\_distance # Updates the value corresponding to the key called "key" to be the value stored in "new\_distance".

path\_taken**[**key**]** **=** current\_node # Adds a new key-value pair to the "path\_taken" dictionary which has key of "key" and value of "current\_node". This is used to show that, in order to get to the station "key", you need to travel to "current\_node" first.

**if** self**.**stepfree **==** **False:** # If "self.stepfree" is False, then run the following lines of code.

end\_node**,**shortest\_distance **=** self**.**find\_end\_node**(**end\_node**,**distances**)** # Calls the "find\_end\_node" function with the parameters "end\_node" and "distances" and stores the returned values in "end\_node" and "shortest\_distance".

path\_list **=** **[**end\_node**]** # Assigns the value of the "end\_node" as the first element in the list called "path\_list".

path\_list **=** self**.**recursive\_find\_path**(**start\_node**,**end\_node**,**path\_taken**,**path\_list**)** # Calls the "recursive\_find\_path" function with the parameters "start\_node", "end\_node", "path\_taken",and "path\_list" and stored the return value in "path\_list".

path\_list**.**reverse**()** # Reverses the order of elements of the list called "path\_list".

**if** self**.**stepfree **==** **False:** # If "self.stepfree" is False, then run the following lines of code.

**return** shortest\_distance**,**path\_list # Returns the variable called "shortest\_distance" and the list called "path\_list".

**else:** # If "self.stepfree" is True, then run the following lines of code.

**return** distances**,**path\_list**,**end\_node # Returns the dictionary called "distances", the list called "path\_list", and the variable called "end\_node".

**def** linear\_search**(**self**,**target**,**items**):** # Defines the "linear\_search" method which takes 2 arguments (other than the object parameter).

# This method is used to find an element in an unordered list (performs a linear search).

length\_of\_list **=** **len(**items**)** # Stores the length of the parameter (and list) called "items" in the variable called "length\_of\_list".

**for** i **in** **range(**length\_of\_list**):** # For loop which is used to iterate through the "items" list as it iterates the same number of times that the length of "items" is.

**if** items**[**i**]** **==** target**:** # If statement: if the element of the list we are iterating through ("items") is the same as the parameter called "target", then run the following code.

**return** i # Returns i which is the index position of the target item ("target") within the list ("items").

**return** **-**1 # If the for loop has run and the if statement within it was never run, then "target" is not in "items", so return -1 to indicate that there is no output for the given inputs (parameters).

**def** find\_end\_node**(**self**,**end\_node**,**distances**):** # Defines the "find\_end\_node" method which takes 2 arguments (other than the object parameter).

# Creates an empty list called "end\_stations\_list" (will append values to this later).

# For every station in "self.stations\_list", if the end\_node is in this, then append it to "end\_stations\_list".

# For every station in the list called "end\_stations\_list", calculate which station has the shortest distance in the "distances" dictionary. Store the station name in "closest\_substation" and the distance in "shortest\_distance".

# Return the closest node ("end\_node" and the shortest distance ("shortest\_distance").

end\_stations\_list **=** **[]** # Creates an empty list called "end\_stations\_list".

**for** i **in** **range** **(**self**.**num\_of\_stations**):** # For loop which iterates through the elements in "self.stations\_list".

key **=** self**.**stations\_list**[**i**]** # Assigns "self.stations\_list[i]" to "key".

**if** end\_node **in** key**:** # If "end\_node" is in the "key", then run the following lines of code.

end\_stations\_list**.**append**(**key**)** # Append "key" in "end\_stations\_list".

**for** i **in** **range** **(len(**end\_stations\_list**)):** # For loop which iterates through each station in "end\_station\_list".

**if** i**==**0**:** # If i is equal to 0, run the following lines of code.

closest\_substation **=** end\_stations\_list**[**0**]** # Assigns the first element of "end\_stations\_list" to the variable called "closest\_substation".

index **=** self**.**linear\_search**(**end\_stations\_list**[**0**],**self**.**stations\_list**)** # Calls the "linear\_search" function with the parameters "end\_stations\_list[0]" and "self.stations\_list", stores the returned value in "index".

shortest\_distance **=** distances**[**closest\_substation**]** # Assigns the value corresponding to the key "closest\_substation" in the dictionary "distances" to the variable called "shortest\_distance".

**else:** # If i is not equal to 0, run the following lines of code.

substation\_key **=** end\_stations\_list**[**i**]** # Assigns "end\_stations\_list[i]" to the variable "substation\_key".

index **=** self**.**linear\_search**(**end\_stations\_list**[**i**],**self**.**stations\_list**)** # Calls the "linear\_search" function with the parameters "end\_stations\_list[i]" and "self.stations\_list", stores the returned value in "index".

**if** distances**[**substation\_key**]** **<** shortest\_distance**:** # If the value in "shortest\_distance" is greater than the value corresponding to the key "substation\_key" in the dictionary "distances",

closest\_substation **=** substation\_key # Assigns the value in "substation\_key" to the variable "closest\_substation".

shortest\_distance **=** distances**[**closest\_substation**]** # Assigns the valeu corresponding to the key "closest\_substation" in the dictionary "distances" in the variable "shortest\_distance".

end\_node **=** closest\_substation # Assigns the value of "closest\_substation" to the variable "end\_node".

**return** end\_node**,**shortest\_distance # Returns the variable "end\_node" and the variable "shortest\_distance".

**def** recursive\_find\_path**(**self**,**start\_node**,**end\_node**,**path\_dict**,**path\_list**):** # Defines the "recursive\_find\_path" method which takes 4 arguments (other than the object parameter).

# This method is used to determine the route that the user should take. It is called wirthin the "dijkstra" function.

# It takes the "path\_dict" and "path\_list" parameters and uses recursion.

# The general case runs if the path\_dict[end\_node] is not the start\_node.

# The base case runs if the path\_dict[end\_node] is the start\_node because this means that the route has been traced back from the end\_node to the start\_node.

# If the general case runs, the value corresponding to path\_dict[end\_node] is appended to path\_list and then the function calls itself, but the new end\_node parameter is the value we just appended.

# Once the base case runs, the path\_list will be a list of the stations that the user should visit, but in reverse order (i.e end node to start node).

**try:** # Try block for error handling.

**if** path\_dict**[**end\_node**]** **==** start\_node**:** # If the value stored in path\_dict[end\_node] is the same as the start\_node, then run the following code.

path\_list**.**append**(**path\_dict**[**end\_node**])** # Append the value stored in path\_dict[end\_node] to "path\_list" (which is a list of all of the stations in the route).

**return** path\_list # Returns "path\_list" which is a list containing elements which are stations in the route the user should take.

**else:** # If the value stored in path\_dict[end\_node] is not the same as the start\_node, then run the following code.

next\_node **=** path\_dict**[**end\_node**]** # Stores the value in path\_dict[end\_node] in the variable called "next\_node".

path\_list**.**append**(**next\_node**)** # Append the value stored in "next\_node" to "path\_list" (which is a list of all of the stations in the route).

self**.**recursive\_find\_path**(**start\_node**,**next\_node**,**path\_dict**,**path\_list**)** # The function calls itself (uses recursion) but the parameter "end\_node" is now the variable "next\_node".

**return** path\_list # Returns "path\_list" which is a list containing elements which are stations in the route the user should take.

**except:** # Except block for error handling.

**pass** # If there is an error, this line is run so that the error is handled.

**def** correct\_lines**(**self**,**line**):** # Defines the "correct\_lines" method which takes 1 argument (other than the object parameter).

# This method is used to adjust the text sored in the "line" parameter in order to be a string which can be outputted for the user.

# This method uses if-elif statements to check what the value of "line" is and creates a variable called "line\_text" which changes depending on what "line" is.

**if** line **==** "CircleLine"**:**

line\_text **=** "(Circle Line)"

**elif** line **==**"CircleLineW"**:**

line\_text **=** "(Circle Line - West)"

**elif** line **==** "CircleLineE"**:**

line\_text **=** "(Circle Line - East)"

**elif** line **==** "CentralLine"**:**

line\_text **=** "(Central Line)"

**elif** line **==** "DistrictLine"**:**

line\_text **=** "(District Line)"

**elif** line **==**"DistrictLineW"**:**

line\_text **=** "(District Line - West)"

**elif** line **==** "DistrictLineE"**:**

line\_text **=** "(District Line - East)"

**elif** line **==** "VictoriaLine"**:**

line\_text **=** "(Victoria Line)"

**elif** line **==** "PiccadillyLine"**:**

line\_text **=** "(Piccadilly Line)"

**elif** line **==** "ElizabethLine"**:**

line\_text **=** "(Elizabeth Line)"

**elif** line **==** "WaterlooLine"**:**

line\_text **=** "(Waterloo & City Line)"

**elif** line **==** "HammersmithLine"**:**

line\_text **=** "(Hammersmith & City Line)"

**elif** line **==**"HammersmithLineW"**:**

line\_text **=** "(Hammersmith & City Line - West)"

**elif** line **==** "HammersmithLineE"**:**

line\_text **=** "(Hammersmith & City Line - East)"

**elif** line **==** "JubileeLine"**:**

line\_text **=** "(Jubilee Line)"

**elif** line **==** "NorthernLine"**:**

line\_text **=** "(Northern Line)"

**elif** line **==** "BakerlooLine"**:**

line\_text **=** "(Bakerloo Line)"

**return** line\_text # Returns the variable called "line\_text" which was created in the function.

**def** main**(**self**):** # Defines the "main" method which takes no arguments (other than the object parameter).

# This function is used to run all of the methods in the class sequentially.

# It corrects station names and creates empty lists.

# It then finds all of the lines that can be used at the start of the journey and therefore finds all possible start station keys for the adjacency dictionary.

# It then runs the "dijkstra" function for all of the possible lines that the user can use at the start station by using a for loop.

# Then, the start station which results in the shortest route is determined and this route is used.

# The "output\_everything" function is run and the route is displayed for the user.

self**.**correct\_stations**()** # Calls the "correct\_stations" function which is used to adjust any station names which are problematic.

possible\_starts **=** **[]** # Creates an empty list called "possible\_starts".

all\_shortest\_distances **=** **[]** # Creates an empty list called "all\_shortest\_distances".

all\_path\_lists **=** **[]** # Creates an empty list called "all\_path\_lists".

**for** i **in** **range(**self**.**num\_of\_stations**):** # For loop which iterates through all of the stations in "stations\_list".

**if** self**.**station1 **in** self**.**stations\_list**[**i**]:** # Checks if "self.station1" (the start station) is the same as "self.stations\_list[i]" which is the element that is being iterated through in the loop.

possible\_starts**.**append**(**self**.**stations\_list**[**i**])** # Appends "self.stations\_list[i]" to the list called "possible\_starts". This list will contain all possible start station keys for the adjacency dictionary.

**for** i **in** **range** **(len(**possible\_starts**)):** # For loop which iterates through all of the elements in the "possible\_starts" list. This for loop performs the "dijkstra" function for all of the possible start stations.

start **=** possible\_starts**[**i**]** # Assigns "possible\_starts[i]" to the variable called "start".

end **=** self**.**station2 # Assigns "self.station2" to the variable called "end".

shortest\_distance**,**path\_list **=** self**.**dijkstra**(**start**,**end**)** # Calls the "dijkstra" function using the parameters "start" and "end" and stores the returned values within "shortest\_distance" , and "path\_list", .

all\_shortest\_distances**.**append**(**shortest\_distance**)** # Appends the value within the variable called "shortest\_distance" to a list called "all\_shortest\_distances".

all\_path\_lists**.**append**(**path\_list**)** # Appends the list within "path\_list" to a list called "all\_path\_lists".

**for** i **in** **range(len(**all\_shortest\_distances**)):** # For loop which iterates through all of the elements within the "all\_shortest\_distances" list.

**if** i **==** 0**:** # If i is 0, then run the following lines of code.

shortest **=** all\_shortest\_distances**[**0**]** # Assigns the value in "all\_shortest\_distances[0]" to the variable called "shortest".

shortest\_index **=** 0 # Assigns the value of 0 to the variable called "shortest\_index".

**else:** # If i is not 0, then run the following lines of code.

**if** all\_shortest\_distances**[**i**]** **<** shortest**:** # If the value stored in "all\_shortest\_distances[i]" is less than the value stored in "shortest", then run the following lines of code.

shortest **=** all\_shortest\_distances**[**i**]** # Assigns the value in "all\_shortest\_distances[i]" to the variable called 'shortest".

shortest\_index **=** i # Assigns the value of i to the variable called "shortest\_index".

path\_list **=** all\_path\_lists**[**shortest\_index**]** # Stores "all\_path\_lists[shortest\_index]" using the identifier called "path\_list".

distance **=** shortest # Stores the value corresponding to the variable "shortest" using the identifier called "distance". This is the shortest time taken to reach the end node from the start node.

self**.**output\_everything**(**path\_list**,**distance**)** # Calls the "output\_everything" method with the parameters "path\_list" and "distance". This is used to output the whole route and distance in html for the user.

**def** output\_everything**(**self**,**path\_list**,**distance**):** # Defines the "output\_everything" method which takes 2 arguments (other than the object parameter).

# This method is used to create a structure for how the route is outputted for the user.

# It starts off by using a for loop to reduce all of the elements in "path\_list" to 2 parts: station and line.

# Then, it creates a new variable for changing the value of the line into a string which is output for the user by calling the "self.correct\_lines" method.

# After this, the station part is changed to a string which is more readable for the user if it is required for the special stations (these may have line names in them, so they are different to the other stations).

# The station and line are concatenated to create a string containing the station and line together which may be outputted later. This is appended to a list (which was empty before the loop).

# Another for loop is used consequently to output each station and line as a html line for the user to see on the webpage.

# Outside of the for loops, there are print statements to show the user the amount of time taken for the journey and links for the user to return to the previous page if they wish to.

# Objectives 8, 8.1, 8.2:

texts **=** **[]** # Creates an empty list called "texts".

**for** i **in** **range(len(**path\_list**)):** # For loop iterates the same number of times as the length of the list called "path\_list".

item **=** path\_list**[**i**]** # Assigns the iterated element of path\_list to the variable called "item".

item\_list **=** item**.**split**()** # Uses built-in function called "split" to separate the words (station and line) in the variable called "item" and stores this in the list called "item\_list".

line\_used **=** item\_list**[-**1**]** # Assigns the last element of the list called "item\_list" to the variable called "line\_used".

item\_list**.**pop**(-**1**)** # The last element of the list called "item\_list" is popped/removed.

**if** line\_used **!=** "EXIT"**:** # If statement to ensure that if the "line\_used" variable is not "EXIT", then run the following lines of code.

line\_text **=** self**.**correct\_lines**(**line\_used**)** # Since the "line\_used" is not "EXIT", "line\_used" is a valid line. Therefore, the method called "self.correct\_lines" is called with the argument "line\_used" and the returned value is stored in the variable called "line\_text".

**else:** # If the "line\_used" is "EXIT", then there is no line used for this part of the route, so run the following lines of code.

line\_text **=** "(Street)" # Assigns a string to the variable "line\_text" to indicate that this part of the route involves reaching the street and not using the tube again.

text **=** "" # Assigns an empty string to the variable called "text".

**for** j **in** **range(len(**item\_list**)):** # For loop: iterates through the list called "item\_list" in order to change station names if necessary and to add a space (" ") at the end of the station name.

element **=** item\_list**[**j**]** # Stores the value in item\_list[j] in the variable called "element".

# If-elif statements to change the names of stations if they are ones which contain other station names or line names within them.

**if** element **==** "VictoriaStation"**:**

element **=** "Victoria"

**elif** element **==** "PiccadillyCircus"**:**

element **=** "Piccadilly Circus"

**elif** element **==** "WaterlooStation"**:**

element **=** "Waterloo"

**elif** element **==** "EustonStation"**:**

element **=** "Euston"

**elif** element **==** "AldgateStation"**:**

element **=** "Aldgate"

text **+=** element **+** " " # Concatenates the variable called "text" with the variable called "element" and a space (" ") and stoes it in the variable called "text".

text **+=** line\_text # Concatenates the variable called "text" with the variable called "line\_text" and stores it in the variable called "text".

texts**.**append**(**text**)** # Appends the value of "text" to the list called "texts".

**for** i **in** **range(len(**texts**)):** # For loop which iterates through the list called "texts" (which contains elements which are the station and line together for the route).

**if** i **!=** **len(**texts**)-**1**:** # If i is not equal to the lenght of the list - 1, then run the folowing code.

texts**[**i**]** **=** texts**[**i**]** **+** " -->" # Appends "-->" to the string stored in texts[i] and stores this within texts[i].

**print(**"<h2> %s</h2>" **%** texts**[**i**])** # html line: outputs the station and line that the user needs to visit as part of the route.

text2 **=** "Approximated Time: " **+** **str(round(**distance **/** 60**))** **+** " minutes" # Creates a string which is used to output the estimated time that the journey will take.

**print(**"<h3> %s</h3>" **%** text2**)** # html line: outputs the time taken for the user to complete the journey.

**print(**"<h3> Want to plan another journey? </h3>"**)** # html line: outputs text on the webpage.

**print** **(**"<a href='../journeyplanner.html'>Journey Planner <a>"**)** # html line: outputs a link for the user to use if they want to return to the previous webpage.

**class** **StepFreeJourneyPlanner(**JourneyPlanner**):** # Creates a class for the step free journey planner called "StepFreeJourneyPlanner" which inherits the methods and attributes from the "JourneyPlanner" class.

**def** \_\_init\_\_**(**self**,**station1**,**station2**,**rush\_hour**,**weekend**):** # Defines the constructor method which takes 4 arguments (other than the object parameter).

# This method is used to define class attributes.

self**.**station1 **=** station1 # Assigns value stored in the parameter "station1" as the "self.station1" attribute.

self**.**station2 **=** station2 # Assigns value stored in the parameter "station2" as the "self.station2" attribute.

self**.**rush\_hour **=** rush\_hour # Assigns value stored in the parameter "rush\_hour" as the "self.rush\_hour" attribute.

self**.**weekend **=** weekend # Assigns value stored in the parameter "weekend" as the "self.weekend" attribute.

# Objectives 1.3, 1.4, 1.5:

# Objectives 2.1, 2.2:

# Objective 7, 7.2:

# Creates an adjacency dictionary attribute called "self.stations\_adjacency\_dict" containing keys which represent the station you are at, and values which are dictionaries as well.

# These dictionaries contain keys which represent the station you can travel to, and the values are the number of seconds taken to travel there.

# The difference between this adjacency dictionary and the one from the "JourneyPlanner" class is that this one has limited access, so the adjacency dictionary may have some stations where you can only travel east on a certain line, or some stations that cannot be accessed, etc.

self**.**stations\_adjacency\_dict**=** **{**

"Paddington CircleLineE"**:** **{**"Edgware Road CircleLine"**:**110.8**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington DistrictLineE"**:** 300**,**"Paddington HammersmithLine"**:**300**,**"Paddington EXIT"**:**300**},**

"Paddington DistrictLineE"**:** **{**"Edgware Road DistrictLine"**:**110.8**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington CircleLineE"**:** 300**,**"Paddington HammersmithLine"**:**300**,**"Paddington EXIT"**:**300**},**

"Paddington CircleLineW"**:** **{**"South Kensington CircleLine"**:**487.4**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington DistrictLineW"**:** 300**,**"Paddington HammersmithLine"**:**300**},**

"Paddington DistrictLineW"**:** **{**"Earl's Court DistrictLine"**:**545.5**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington CircleLineW"**:** 300**,**"Paddington HammersmithLine"**:**300**},**

"Paddington HammersmithLine"**:{**"Edgware Road HammersmithLine"**:**110.8**,**"Paddington BakerlooLine"**:**300**,**"Paddington ElizabethLine"**:** 300**,**"Paddington CircleLineE"**:** 300**,**"Paddington DistrictLineE"**:**300**,**"Paddington EXIT"**:**300**},**

"Paddington BakerlooLine"**:** **{**"Baker Street BakerlooLine"**:**208.7**,** "Paddington ElizabethLine"**:** 300**,**"Paddington DistrictLineE"**:** 300**,**"Paddington CircleLineE"**:** 300**,**"Paddington HammersmithLine"**:**300**,**"Paddington EXIT"**:**300**},**

"Paddington ElizabethLine"**:** **{**"Tottenham Court Road ElizabethLine"**:**241.6**,**"Paddington BakerlooLine"**:**300**,**"Paddington DistrictLineE"**:** 300**,**"Paddington CircleLineE"**:** 300**,**"Paddington HammersmithLine"**:**300**},**

"Paddington EXIT"**:** **{**"Paddington ElizabethLine"**:**300**,**"Paddington BakerlooLine"**:**300**,**"Paddington HammersmithLine"**:**300**,**"Paddington DistrictLineE"**:**300**,**"Paddington CircleLineE"**:**300**},**

"Earl's Court DistrictLine"**:** **{**"Paddington DistrictLineE"**:**545.5**,** "South Kensington DistrictLine"**:**175.6**,** "Earl's Court PiccadillyLine"**:**300**,**"Earl's Court EXIT"**:**300**},**

"Earl's Court PiccadillyLine"**:** **{**"Green Park PiccadillyLine"**:**511.9**,** "Earl's Court DistrictLine"**:**300**,**"Earl's Court EXIT"**:**300**},**

"Earl's Court EXIT"**:** **{**"Earl's Court DistrictLine"**:**300**,**"Earl's Court PiccadillyLine"**:**300**},**

"South Kensington CircleLine"**:** **{**"VictoriaStation CircleLine"**:**199.1**,**"Paddington CircleLineE"**:**487.4**,**"South Kensington DistrictLine"**:**300**,**"South Kensington EXIT"**:**300**},**

"South Kensington DistrictLine"**:** **{**"VictoriaStation DistrictLine"**:**199.1**,**"South Kensington CircleLine"**:**300**,**"South Kensington EXIT"**:**300**},**

"South Kensington EXIT"**:** **{**"South Kensington CircleLine"**:**300**,**"South Kensington DistrictLine"**:**300**},**

"VictoriaStation CircleLine"**:** **{**"South Kensington CircleLine"**:**199.1**,**"Westminster CircleLine"**:**161.2**,**"VictoriaStation DistrictLine"**:**300**,**"VictoriaStation VictoriaLine"**:**300**,**"VictoriaStation EXIT"**:**300**},**

"VictoriaStation DistrictLine"**:** **{**"South Kensington DistrictLine"**:**199.1**,**"Westminster DistrictLine"**:**161.2**,**"VictoriaStation CircleLine"**:**300**,**"VictoriaStation VictoriaLine"**:**300**,**"VictoriaStation EXIT"**:**300**},**

"VictoriaStation VictoriaLine"**:** **{**"Vauxhall VictoriaLine"**:**144.4**,**"Green Park VictoriaLine"**:**78.4**,**"VictoriaStation CircleLine"**:**300**,**"VictoriaStation DistrictLine"**:**300**,**"VictoriaStation EXIT"**:**300**},**

"VictoriaStation EXIT"**:** **{**"VictoriaStation CircleLine"**:**300**,**"VictoriaStation DistrictLine"**:**300**,**"VictoriaStation VictoriaLine"**:**300**},**

"Westminster CircleLine"**:** **{**"VictoriaStation CircleLine"**:**161.2**,**"Blackfriars CircleLine"**:**234.6**,**"Westminster DistrictLine"**:**300**,**"Westminster JubileeLine"**:**300**,**"Westminster EXIT"**:**300**},**

"Westminster DistrictLine"**:** **{**"VictoriaStation DistrictLine"**:**161.2**,**"Blackfriars DistrictLine"**:**234.6**,**"Westminster CircleLine"**:**300**,**"Westminster JubileeLine"**:**300**,**"Westminster EXIT"**:**300**},**

"Westminster JubileeLine"**:** **{**"WaterlooStation JubileeLine"**:**78.1**,**"Green Park JubileeLine"**:**99.8**,**"Westminster DistrictLine"**:**300**,**"Westminster CircleLine"**:**300**,**"Westminster EXIT"**:**300**},**

"Westminster EXIT"**:** **{**"Westminster DistrictLine"**:**300**,**"Westminster CircleLine"**:**300**,**"Westminster JubileeLine"**:**300**},**

"Blackfriars CircleLine"**:** **{**"Cannon Street CircleLineE"**:**152.8**,**"Westminster CircleLine"**:**234.6**,**"Blackfriars DistrictLine"**:**300**,**"Blackfriars EXIT"**:**300**},**

"Blackfriars DistrictLine"**:** **{**"Cannon Street DistrictLineE"**:**152.8**,**"Westminster DistrictLine"**:**234.6**,**"Blackfriars CircleLine"**:**300**,**"Blackfriars EXIT"**:**300**},**

"Blackfriars EXIT"**:** **{**"Blackfriars CircleLine"**:**300**,**"Blackfriars DistrictLine"**:**300**},**

"Cannon Street CircleLineW"**:** **{**"Blackfriars CircleLine"**:**152.8**,**"Cannon Street DistrictLineW"**:**300**,**"Cannon Street EXIT"**:**300**},**

"Cannon Street DistrictLineW"**:** **{**"Blackfriars DistrictLine"**:**152.8**,**"Cannon Street CircleLineW"**:**300**,**"Cannon Street EXIT"**:**300**},**

"Cannon Street CircleLineE"**:** **{**"Tower Hill CircleLine"**:**158.8**,**"Cannon Street DistrictLineE"**:**300**},**

"Cannon Street DistrictLineE"**:** **{**"Tower Hill DistrictLine"**:**158.8**,**"Cannon Street CircleLineE"**:**300**},**

"Cannon Street EXIT"**:** **{**"Cannon Street CircleLineW"**:**300**,**"Cannon Street DistrictLineW"**:**300**},**

"Tower Hill CircleLine"**:** **{**"Cannon Street CircleLineW"**:**158.8**,**"Liverpool Street CircleLineW"**:**198.8**,**"Tower Hill DistrictLine"**:**300**,**"Tower Hill EXIT"**:**300**},**

"Tower Hill DistrictLine"**:** **{**"Cannon Street DistrictLineW"**:**158.8**,**"Aldgate East DistrictLine"**:**132.9**,**"Tower Hill CircleLine"**:**300**,**"Tower Hill EXIT"**:**300**},**

"Tower Hill EXIT"**:** **{**"Tower Hill CircleLine"**:**300**,**"Tower Hill DistrictLine"**:**300**},**

"Aldgate East DistrictLine"**:** **{**"Tower Hill DistrictLine"**:**132.9**,**"Aldgate East HammersmithLine"**:**300**,**"Aldgate East EXIT"**:**300**},**

"Aldgate East HammersmithLine"**:** **{**"Liverpool Street HammersmithLineW"**:**128.4**,**"Aldgate East DistrictLine"**:**300**,**"Aldgate East EXIT"**:**300**},**

"Aldgate East EXIT"**:** **{**"Aldgate East DistrictLine"**:**300**,**"Aldgate East HammersmithLine"**:**300**},**

"Liverpool Street CircleLineE"**:** **{**"Tower Hill CircleLine"**:**198.8**,**"Liverpool Street HammersmithLineE"**:**300**,**"Liverpool Street ElizabethLine"**:**300**,**"Liverpool Street EXIT"**:**300**},**

"Liverpool Street HammersmithLineE"**:** **{**"Aldgate East HammersmithLine"**:**128.4**,**"Liverpool Street CircleLineE"**:**300**,**"Liverpool Street ElizabethLine"**:**300**,**"Liverpool Street EXIT"**:**300**},**

"Liverpool Street CircleLineW"**:** **{**"Moorgate CircleLine"**:**70.6**,**"Liverpool Street HammersmithLineW"**:**300**,**"Liverpool Street ElizabethLine"**:**300**},**

"Liverpool Street HammersmithLineW"**:** **{**"Moorgate HammersmithLine"**:**70.6**,**"Liverpool Street CircleLineW"**:**300**,**"Liverpool Street ElizabethLine"**:**300**},**

"Liverpool Street ElizabethLine"**:** **{**"Farringdon ElizabethLine"**:**96.0**,** "Liverpool Street CircleLineE"**:**300**,**"Liverpool Street HammersmithLineE"**:**300**,**"Liverpool Street EXIT"**:**300**},**

"Liverpool Street EXIT"**:** **{**"Liverpool Street CircleLineE"**:**300**,**"Liverpool Street HammersmithLineE"**:**300**,**"Liverpool Street ElizabethLine"**:**300**},**

"Moorgate CircleLine"**:** **{**"Liverpool Street CircleLineE"**:**70.6**,**"Barbican CircleLineW"**:**72.1**,**"Moorgate HammersmithLine"**:**300**,**"Moorgate NorthernLine"**:**300**,**"Moorgate EXIT"**:**300**},**

"Moorgate HammersmithLine"**:** **{**"Liverpool Street HammersmithLineE"**:**70.6**,**"Barbican HammersmithLineW"**:**72.1**,**"Moorgate CircleLine"**:**300**,**"Moorgate NorthernLine"**:**300**,**"Moorgate EXIT"**:**300**},**

"Moorgate NorthernLine"**:** **{**"King's Cross St. Pancras NorthernLine"**:**341.6**,**"London Bridge NorthernLine"**:**168.3**,**"Moorgate CircleLine"**:**300**,**"Moorgate HammersmithLine"**:**300**,**"Moorgate EXIT"**:**300**},**

"Moorgate EXIT"**:** **{**"Moorgate CircleLine"**:**300**,**"Moorgate HammersmithLine"**:**300**,**"Moorgate NorthernLine"**:**300**},**

"Barbican CircleLineE"**:** **{**"Moorgate CircleLine"**:**72.1**,**"Barbican HammersmithLineE"**:**300**},**

"Barbican HammersmithLineE"**:** **{**"Moorgate HammersmithLine"**:**72.1**,**"Barbican CircleLineE"**:**300**},**

"Barbican CircleLineW"**:** **{**"Farringdon CircleLine"**:**67.6**,**"Barbican HammersmithLineW"**:**300**,**"Barbican EXIT"**:**300**},**

"Barbican HammersmithLineW"**:** **{**"Farringdon HammersmithLine"**:**67.6**,**"Barbican CircleLineW"**:**300**,**"Barbican EXIT"**:**300**},**

"Barbican EXIT"**:** **{**"Barbican CircleLineW"**:**300**,**"Barbican HammersmithLineW"**:**300**},**

"Farringdon ElizabethLine"**:** **{**"Liverpool Street ElizabethLine"**:**96.0**,**"Tottenham Court Road ElizabethLine"**:**138.8**,**"Farringdon CircleLine"**:**300**,**"Farringdon HammersmithLine"**:**300**,**"Farringdon EXIT"**:**300**},**

"Farringdon CircleLine"**:** **{**"Barbican CircleLineE"**:**67.6**,**"King's Cross St. Pancras CircleLine"**:**164.0**,**"Farringdon ElizabethLine"**:**300**,**"Farringdon HammersmithLine"**:**300**,**"Farringdon EXIT"**:**300**},**

"Farringdon HammersmithLine"**:** **{**"Barbican HammersmithLineE"**:**67.6**,**"King's Cross St. Pancras HammersmithLine"**:**164.0**,**"Farringdon ElizabethLine"**:**300**,**"Farringdon CircleLine"**:**300**,**"Farringdon EXIT"**:**300**},**

"Farringdon EXIT"**:** **{**"Farringdon ElizabethLine"**:**300**,**"Farringdon CircleLine"**:**300**,**"Farringdon HammersmithLine"**:**300**},**

"King's Cross St. Pancras CircleLine"**:** **{**"Farringdon CircleLine"**:**164.0**,**"Euston Square CircleLineW"**:**83.2**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**,**"King's Cross St. Pancras EXIT"**:**300**},**

"King's Cross St. Pancras HammersmithLine"**:** **{**"Farringdon HammersmithLine"**:**164.0**,**"Euston Square HammersmithLineW"**:**83.2**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**,**"King's Cross St. Pancras EXIT"**:**300**},**

"King's Cross St. Pancras NorthernLine"**:** **{**"Moorgate NorthernLine"**:**341.6**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**,**"King's Cross St. Pancras EXIT"**:**300**},**

"King's Cross St. Pancras VictoriaLine"**:** **{**"EustonStation VictoriaLine"**:**87.4**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**,**"King's Cross St. Pancras EXIT"**:**300**},**

"King's Cross St. Pancras PiccadillyLine"**:** **{**"Green Park PiccadillyLine"**:**441.4**,**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras EXIT"**:**300**},**

"King's Cross St. Pancras EXIT"**:** **{**"King's Cross St. Pancras HammersmithLine"**:**300**,**"King's Cross St. Pancras NorthernLine"**:**300**,**"King's Cross St. Pancras VictoriaLine"**:**300**,**"King's Cross St. Pancras CircleLine"**:**300**,**"King's Cross St. Pancras PiccadillyLine"**:**300**},**

"Euston Square CircleLineE"**:** **{**"King's Cross St. Pancras CircleLine"**:**83.2**,**"Euston Square HammersmithLineE"**:**300**},**

"Euston Square HammersmithLineE"**:** **{**"King's Cross St. Pancras HammersmithLine"**:**83.2**,**"Euston Square CircleLineE"**:**300**},**

"Euston Square CircleLineW"**:** **{**"Edgware Road CircleLine"**:**110.8**,**"Euston Square HammersmithLineW"**:**300**,**"Euston Square EXIT"**:**300**},**

"Euston Square HammersmithLineW"**:** **{**"Edgware Road HammersmithLine"**:**110.8**,**"Euston Square CircleLineW"**:**300**,**"Euston Square EXIT"**:**300**},**

"Euston Square EXIT"**:** **{**"Euston Square CircleLineW"**:**300**,**"Euston Square HammersmithLineW"**:**300**},**

"Baker Street BakerlooLine"**:** **{**"Oxford Circus BakerlooLine"**:**204.7**,**"Paddington BakerlooLine"**:**208.7**,**"Baker Street JubileeLine"**:**300**,**"Baker Street EXIT"**:**300**},**

"Baker Street JubileeLine"**:** **{**"Bond Street JubileeLine"**:**112.5**,**"Baker Street BakerlooLine"**:**300**,**"Baker Street EXIT"**:**300**},**

"Baker Street EXIT"**:** **{**"Baker Street JubileeLine"**:**300**,**"Baker Street BakerlooLine"**:**300**},**

"Edgware Road CircleLine"**:** **{**"Euston Square CircleLineE"**:**265.0**,**"Paddington CircleLineW"**:**110.8**,**"Edgware Road HammersmithLine"**:**300**,**"Edgware Road DistrictLine"**:**300**,**"Edgware Road EXIT"**:**300**},**

"Edgware Road HammersmithLine"**:** **{**"Euston Square HammersmithLineE"**:**265.0**,**"Paddington HammersmithLine"**:**110.8**,**"Edgware Road CircleLine"**:**300**,**"Edgware Road DistrictLine"**:**300**,**"Edgware Road EXIT"**:**300**},**

"Edgware Road DistrictLine"**:** **{**"Paddington DistrictLineW"**:**93.1**,**"Edgware Road HammersmithLine"**:**300**,**"Edgware Road CircleLine"**:**300**,**"Edgware Road EXIT"**:**300**},**

"Edgware Road EXIT"**:** **{**"Edgware Road HammersmithLine"**:**300**,**"Edgware Road CircleLine"**:**300**,**"Edgware Road DistrictLine"**:**300**},**

"EustonStation VictoriaLine"**:** **{**"King's Cross St. Pancras VictoriaLine"**:**87.4**,**"Oxford Circus VictoriaLine"**:**130.2**,**"EustonStation EXIT"**:**300**},**

"EustonStation EXIT"**:** **{**"EustonStation VictoriaLine"**:**300**},**

"Bond Street CentralLine"**:** **{**"Tottenham Court Road CentralLine"**:**153.6**,**"Bond Street JubileeLine"**:**300**,**"Bond Street EXIT"**:**300**},**

"Bond Street JubileeLine"**:** **{**"Baker Street JubileeLine"**:**112.5**,**"Green Park JubileeLine"**:**94.3**,**"Bond Street CentralLine"**:**300**,**"Bond Street EXIT"**:**300**},**

"Bond Street EXIT"**:** **{**"Bond Street CentralLine"**:**300**,**"Bond Street JubileeLine"**:**300**},**

"Green Park JubileeLine"**:** **{**"Bond Street JubileeLine"**:**94.3**,**"Westminster JubileeLine"**:**99.8**,**"Green Park PiccadillyLine"**:**300**,**"Green Park VictoriaLine"**:**300**,**"Green Park EXIT"**:**300**},**

"Green Park PiccadillyLine"**:** **{**"Earl's Court PiccadillyLine"**:**511.9**,**"King's Cross St. Pancras PiccadillyLine"**:**441.4**,**"Green Park JubileeLine"**:**300**,**"Green Park VictoriaLine"**:**300**,**"Green Park EXIT"**:**300**},**

"Green Park VictoriaLine"**:** **{**"Oxford Circus VictoriaLine"**:**77.6**,**"VictoriaStation VictoriaLine"**:**78.4**,**"Green Park PiccadillyLine"**:**300**,**"Green Park JubileeLine"**:**300**,**"Green Park EXIT"**:**300**},**

"Green Park EXIT"**:** **{**"Green Park PiccadillyLine"**:**300**,**"Green Park JubileeLine"**:**300**,**"Green Park VictoriaLine"**:**300**},**

"Oxford Circus BakerlooLine"**:** **{**"Baker Street BakerlooLine"**:**204.7**,**"Oxford Circus VictoriaLine"**:**300**,**"Oxford Circus EXIT"**:**300**},**

"Oxford Circus VictoriaLine"**:** **{**"Green Park VictoriaLine"**:**77.6**,**"EustonStation VictoriaLine"**:**130.2**,**"Oxford Circus BakerlooLine"**:**300**,**"Oxford Circus EXIT"**:**300**},**

"Oxford Circus EXIT"**:** **{**"Oxford Circus BakerlooLine"**:**300**,**"Oxford Circus VictoriaLine"**:**300**},**

"Tottenham Court Road CentralLine"**:** **{**"Bond Street CentralLine"**:**153.6**,**"Tottenham Court Road ElizabethLine"**:**300**,**"Tottenham Court Road NorthernLine"**:**300**,**"Tottenham Court Road EXIT"**:**300**},**

"Tottenham Court Road ElizabethLine"**:** **{**"Farringdon ElizabethLine"**:**138.8**,**"Paddington ElizabethLine"**:**241.6**,**"Tottenham Court Road CentralLine"**:**300**,**"Tottenham Court Road NorthernLine"**:**300**,**"Tottenham Court Road EXIT"**:**300**},**

"Tottenham Court Road NorthernLine"**:** **{**"Kennington NorthernLine"**:**325.5**,**"Tottenham Court Road ElizabethLine"**:**300**,**"Tottenham Court Road CentralLine"**:**300**,**"Tottenham Court Road EXIT"**:**300**},**

"Tottenham Court Road EXIT"**:** **{**"Tottenham Court Road ElizabethLine"**:**300**,**"Tottenham Court Road CentralLine"**:**300**,**"Tottenham Court Road NorthernLine"**:**300**,**"Tottenham Court Road EXIT"**:**300**},**

"WaterlooStation JubileeLine"**:** **{**"Westminster JubileeLine"**:**78.1**,**"Southwark JubileeLine"**:**49.5**,**"WaterlooStation EXIT"**:**300**},**

"WaterlooStation EXIT"**:** **{**"WaterlooStation JubileeLine"**:**300**},**

"Southwark JubileeLine"**:** **{**"WaterlooStation JubileeLine"**:**49.5**,**"London Bridge JubileeLine"**:**92.2**,**"Southwark EXIT"**:**300**},**

"Southwark EXIT"**:** **{**"Southwark JubileeLine"**:**300**},**

"London Bridge JubileeLine"**:** **{**"Southwark JubileeLine"**:**92.2**,**"London Bridge NorthernLine"**:**300**,**"London Bridge EXIT"**:**300**},**

"London Bridge NorthernLine"**:** **{**"Moorgate NorthernLine"**:**168.3**,**"Borough NorthernLine"**:**79.0**,**"London Bridge JubileeLine"**:**300**,**"London Bridge EXIT"**:**300**},**

"London Bridge EXIT"**:** **{**"London Bridge JubileeLine"**:**300**,**"London Bridge NorthernLine"**:**300**},**

"Borough NorthernLine"**:** **{**"Elephant & Castle NorthernLine"**:**94.2**,**"London Bridge NorthernLine"**:**79.0**,**"Borough EXIT"**:**300**},**

"Borough EXIT"**:** **{**"Borough NorthernLine"**:**300**},**

"Elephant & Castle NorthernLine"**:** **{**"Borough NorthernLine"**:**94.2**,**"Kennington NorthernLine"**:**100.7**,**"Elephant & Castle EXIT"**:**300**},**

"Elephant & Castle EXIT"**:** **{**"Elephant & Castle NorthernLine"**:**300**},**

"Battersea Power Station NorthernLine"**:** **{**"Nine Elms NorthernLine"**:**98.8**,**"Battersea Power Station EXIT"**:**300**},**

"Battersea Power Station EXIT"**:** **{**"Battersea Power Station NorthernLine"**:**300**},**

"Nine Elms NorthernLine"**:** **{**"Battersea Power Station NorthernLine"**:**98.8**,**"Kennington NorthernLine"**:**168.5**,**"Nine Elms EXIT"**:**300**},**

"Nine Elms EXIT"**:** **{**"Nine Elms NorthernLine"**:**300**},**

"Kennington NorthernLine"**:** **{**"Nine Elms NorthernLine"**:**168.5**,**"Tottenham Court Road NorthernLine"**:**325.5**,**"Elephant & Castle NorthernLine"**:**100.7**,**"Kennington EXIT"**:**300**},**

"Kennington EXIT"**:** **{**"Kennington NorthernLine"**:**300**},**

"Vauxhall VictoriaLine"**:** **{**"VictoriaStation VictoriaLine"**:**144.4**,**"Vauxhall EXIT"**:**300**},**

"Vauxhall EXIT"**:** **{**"Vauxhall VictoriaLine"**:**300**}**

**}**

self**.**stations\_list **=** **list(**self**.**stations\_adjacency\_dict**.**keys**())** # Assigns keys of the stations adjacency dictionary as the "self.stations\_list" attribute.

self**.**num\_of\_stations **=** **len(**self**.**stations\_list**)** # Assigns the length of the stations adjacency dictionary as the "self.num\_of\_stations" attribute.

self**.**stepfree **=** **True** # Assigns the Boolean True value as the "self.stepfree" attribute.

**def** main**(**self**):** # Defines the "main" method which takes no arguments (other than the object parameter). This method overrides the "main" method from the "JourneyPlanner" class.

# This function is used to run all of the methods in the class sequentially.

# It corrects station names and checks if the inputted stations are valid for a step-free journey.

# If the inputs are not valid, then an appropriate error messgae is displayed.

# If the inputs are valid, then the "dijkstra" function is run.

# Then, the "output\_everything" function is run and the route is displayed for the user.

valid\_start **=** **False** # Creates a variable called "valid\_start" and sets it to the Boolean value False.

valid\_end **=** **False** # Creates a variable called "valid\_end" and sets it to the Boolean value False.

**if** self**.**station1 **==** "Aldgate"**:**

self**.**station1 **=** "AldgateStation"

**elif** self**.**station1 **==** "Euston"**:**

self**.**station1 **=** "EustonStation"

**if** self**.**station2 **==** "Aldgate"**:**

self**.**station2 **=** "AldgateStation"

**elif** self**.**station2 **==** "Euston"**:**

self**.**station2 **=** "EustonStation"

**for** i **in** **range(**self**.**num\_of\_stations**):** # For loop to iterate through all of the stations that can be visited in a step-free journey.

key **=** self**.**stations\_list**[**i**]** # Stores the element in the list called "self.stations\_list" corresponding to the index position i in the variable called "key".

**if** self**.**station1 **in** key**:** # If the value within the variable "self.station1" is in key, run the following code.

valid\_start **=** **True** # Sets the variable "valid\_start" to the Boolean value True.

**if** self**.**station2 **in** key**:** # If the value within the variable "self.station2" is in key, run the following code.

valid\_end **=** **True** # Sets the variable "valid\_end" to the Boolean value True.

**if** valid\_start **==** **True** **and** valid\_end **==** **True:** # If "valid\_start" and "valid\_end" are both True (the inputs are valid), then run the following lines of code.

self**.**correct\_stations**()** # Calls the "correct\_stations" function which is used to adjust any station names which are problematic.

start\_node **=** self**.**station1 **+** " EXIT" # Concatenates the string in the variable called "self.station1" with " EXIT" and stores this in the variable called "start\_node".

end\_node **=** self**.**station2 **+** " EXIT" # Concatenates the string in the variable called "self.station2" with " EXIT" and stores this in the variable called "end\_node".

distances**,**path\_list**,**end\_node **=** self**.**dijkstra**(**start\_node**,**end\_node**)** # Calls the "dijkstra" function using the parameters "start\_node" and "end\_node" and stores the returned values within "distances" , "path\_list", and "end\_node".

distance **=** distances**[**end\_node**]** # Stores the value corresponding to the key "end\_node" in the "distances" dictionary using the identifier called "distance". This is the shortest time taken to reach the end node from the start node.

self**.**output\_everything**(**path\_list**,**distance**)** # Calls the "output\_everything" method with the parameters "path\_list" and "distance". This is used to output the whole route and distance in html for the user.

**else:** # If "valid\_start" and "valid\_end" are not True together (the inputs are invalid), then run the following lines of code.

**print(**"<h2> Sorry, you have entered a station without any step free access but have chosen step free mode :( </h2>"**)** # html line: outputs a message on the webpage which tells the user that they have invalid inputs.

**print(**"<h2> You can re-enter a route here: </h2>"**)** # html line: outputs text on the webpage.

**print** **(**"<a href='../journeyplanner.html'>Journey Planner <a>"**)** # html line: outputs a link for the user to use if they want to return to the previous webpage.

#station1="Piccadilly Circus"

#station2="Cannon Street"

#rush\_hour="No"

#weekend="No"

#step\_free = "No"

**if** step\_free **==** "No"**:** # If the user has inputted "No" when asked if they want a step free journey, then the journey will use the normal adjacency dictionary and the normal "JourneyPlanner" class.

jp **=** JourneyPlanner**(**station1**,**station2**,**rush\_hour**,**weekend**)** # Instantiation: creates an object called "jp" using the "JourneyPlanner" class with 4 arguments.

jp**.**main**()** # Calls the method called "main" in the "jp" object.

**else:** # If the user has inputted "Yes" when asked if they want a step free journey, then the journey will use the adjusted adjacency dictionary and the "StepFreeJourneyPlanner" class.

sf\_jp **=** StepFreeJourneyPlanner**(**station1**,**station2**,**rush\_hour**,**weekend**)** # Instantiation: creates an object called "sf\_jp" using the "StepFreeJourneyPlanner" class with 4 arguments.

sf\_jp**.**main**()** # Calls the method called "main" in the "sf\_jp" object.

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<div class='footer'>"**)** # html line: references the footer div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Computer Science: Non-Exam Assessment</h3>"**)** # html line: text within the footer div element is output.

**print(**"<h3>2022/23</h3>"**)** # html line: text within the footer div element is output.

**print(**"</div>"**)** # html line: closes the footer div element defined from the CSS file for structuring the web page.

**print** **(**"</body>"**)** # html line: the closing </body> element for structuring the web page

A screenshot of a computer

Description automatically generated**print** **(**"</html>"**)** # html line: the closing </html> element for structuring the web page

A screenshot of a computer

Description automatically generatedStep free route:

arrivaltimes.pyw:

**import** cgi**,** cgitb**,** shutil**,** glob**,** os # Import module which will be used later in the program.

**import** tubeAPI # Import class which will be used later in the program.

form **=** cgi**.**FieldStorage**()** # Data inputted from the form is stored so that it can be used later in the program.

station **=** form**.**getvalue**(**'station\_arrivals'**)** # The station that the user inputted is accessed from the form and stored as the variable called "station".

# Print statements output as html lines:

**print** **(**"Content-type:text/html\n\n"**)** # html line: defines content type so that the lines are processed and interpretted correctly by the computer.

**print** **(**"<html>"**)** # html line: the <html> element for structuring the web page.

**print** **(**"<head>"**)** # html line: the <head> element for structuring the web page.

**print** **(**"<title>Arrivals At Tube Stations</title>"**)** # html line: the <title> element for distinguishing which page the user is on.

**print(**"<link rel='stylesheet' type='text/css' href='/styles.css'>"**)** # html line: references the file used for CSS styling of the webpage (objective 4.2).

**print(**"<meta name='viewport' content='width=device-width, initial-scale=1'>"**)** # html line: formats the webpage so that the content atomatically adjusts its size to fit the screen.

**print** **(**"</head>"**)** # html line: the closing </head> element for structuring the web page.

**print(**"<body>"**)** # html line: the <body> element for structuring the web page.

# Objective 3.1:

**print(**"<div class='navbar'>"**)** # html line: references the navbar div defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<a href='/journeyplanner.html'class='left'>Journey Planner</a>"**)** # html line: the first part of the navbar is for the journey planner - when clicked, the website redirects to the html for the journey planner page.

**print(**"<a href='/map.html'class='center'>Map</a>"**)** # html line: the second part of the navbar is for the map - when clicked, the website redirects to the html for the map page.

**print(**"<a href='/updates.html'class='center'>Status Updates</a>"**)** # html line: the third part of the navbar is for status updates - when clicked, the website redirects to the html for the status updates page.

**print(**"<a href='/arrivaltimes.html'>Arrival Times</a>"**)** # html line: the fourth part of the navbar is for the arrival times - when clicked, the website redirects to the html for the arrival times page.

**print(**"<a href='/liftupdates.html'>Lift Updates</a>"**)** # html line: the fifth part of the navbar is for the lift updates - when clicked, the website redirects to the html for the lift updates page.

**print(**"<a href='/index.html'class='right'>About</a>"**)** # html line: the last part of the navbar is for the information page - when clicked, the website redirects to the html for the about page.

**print(**"</div>"**)** # html line: closing tag for the navbar div defined from the CSS file for structuring the web page.

**print(**"<div class='header'>"**)** # html line: references the header div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Tube Journey Planner</h3>"**)** # html line: text within the header div element is output.

**print(**"<h3>Farheen Fahim</h3>"**)** # html line: text within the header div element is output.

**print(**"</div>"**)** # html line: closes the header div element defined from the CSS file for structuring the web page.

**print(**"<h1>Arrivals</h1>"**)** # html line: outputs a title for the webpage.

# Objective 3.3:

endpoint **=** **(**"https://api.tfl.gov.uk/Stoppoint/Search/?query=" **+** station **+** "&modes=tube&app\_key=61cab23336c147e29c64f5de382d1d91"**)** # the variable called "endpoint" stores the string which is the url for accessing the API information.

t **=** tubeAPI**.**Tube\_API**()** # Instantiation: creates an object called "t" using the class called "Tube\_API" from the "tubeAPI" file which was imported at the start.

ID**,** ICSCODE **=** t**.**get\_station\_id**(**station**)** # Calls the "get\_station\_id" method within the object called "t" with the argument "station", the values returned are stored using the identifiers called "ID" and "ICSCODE.

endpoint **=** **(**"https://api.tfl.gov.uk/Mode/tube/Arrivals/?app\_key=61cab23336c147e29c64f5de382d1d91"**)** # the variable called "endpoint" stores the string which is the url for accessing the API information.

data **=** t**.**get\_data**(**endpoint**)** # Calls the "get\_data" method within the object called "t" with the argument "endpoint", the value returned is stored using the identifier called "data".

all\_info **=** **[]** # Creates an empty list called "all\_info".

**for** i **in** **range(len(**data**)):** # For loop: iterates through the list called "data".

dictionary **=** data**[**i**]** # Stores the value from the "data" list which has index position i using the identifier called "dictionary".

**if** station **in** dictionary**[**"stationName"**]:** # If the station which the user entered is in the data we are iterating through, then it is relevant, and we need to filter for this data, so run the following line of code.

all\_info**.**append**(**dictionary**)** # Append the relevent information to the (empty) list so that this list only contains the information on the station the user has entered.

**if** **len(**all\_info**)** **==** 0**:** # If the length of the list called "all\_info" is 0, there is no information on any arrivals to this station at the moment, so run the following line of code.

**print(**"<h3> No arrivals at this station at the moment, sorry :( </h3>"**)** # html line: since there is no information on any arrivals to this station at the moment, this line displays a relevant message for the user.

**else:** # If the length of the list called "all\_info" is not 0, then there is information on one or many arrivals at the station the user has entered, so run the following lines of code to display the information accessed from the API.

# Objective 9.1:

**for** i **in** **range(len(**all\_info**)):** # For loop: iterates through the list called "all\_info" so that all arrivals are outputted for the user.

information **=** all\_info**[**i**]** # Stores the value from the "all\_info" list which has index position i using the identifier called "information".

text **=** station # Assigns the value stored in the variable "station" to the variable called "text".

#Creating strings which are ready to be output (involves concatenating strings with strings accessed from the dictionary with information on the arrival):

text0 **=** "Arrival " **+** **str(**i**+**1**)** # Stores a string (which helps to label each arrival when outputting it for the user) in the variable called "text0".

text1 **=** "Line name: " **+** information**[**"lineName"**]** # Stores a string (which contains information about the line name of the arriving train) in the variable called "text1".

text2 **=** "Platform name: " **+** information**[**"platformName"**]** # Stores a string (which contains information about the platform name of the arriving train) in the variable called "text2".

text3 **=** "Direction: " **+** information**[**"direction"**]** # Stores a string (which contains information about the direction of the arriving train) in the variable called "text3".

text4 **=** "Destination name: " **+** information**[**"destinationName"**]** # Stores a string (which contains information about the destination name of the arriving train) in the variable called "text4".

text5 **=** "Current Location: " **+** information**[**"currentLocation"**]** # Stores a string (which contains information about the current location of the arriving train) in the variable called "text5".

text6 **=** "Towards: " **+** information**[**"towards"**]** # Stores a string (which contains information about the station that the arriving train is going towards) in the variable called "text6".

text7 **=** "Expected arrival:" **+** information**[**"expectedArrival"**]** # Stores a string (which contains information about the expected arrival time of the arriving train) in the variable called "text7".

**print(**"<h3> %s</h3>" **%** text**)** # html line: outputs a line which adds a title for which station the user is loking at arrivals for.

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<h3> %s</h3>" **%** text0**)** # html line: outputs a line which orders the data by adding a subtitle for which arrival it is (e.g. arrival 1, 2, 3 etc).

**print(**"<h3> %s</h3>" **%** text1**)** # html line: outputs the data from the API which corresponds to the station that the user entered (which line is used for the arrival).

**print(**"<h3> %s</h3>" **%** text2**)** # html line: outputs the data from the API which corresponds to the station that the user entered (which platform the arrival will be at).

**print(**"<h3> %s</h3>" **%** text3**)** # html line: outputs the data from the API which corresponds to the station that the user entered (the direction the train is travelling).

**print(**"<h3> %s</h3>" **%** text4**)** # html line: outputs the data from the API which corresponds to the station that the user entered (the destination of the train).

**print(**"<h3> %s</h3>" **%** text5**)** # html line: outputs the data from the API which corresponds to the station that the user entered (the train's current location).

**print(**"<h3> %s</h3>" **%** text6**)** # html line: outputs the data from the API which corresponds to the station that the user entered (where the train is going towards).

**print(**"<h3> %s</h3>" **%** text7**)** # html line: outputs the data from the API which corresponds to the station that the user entered (the expected arrival time of the train).

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<h3> Want to look at other station arrivals? </h3>"**)** # html line: outputs text on the webpage.

**print** **(**"<a href='../arrivaltimes.html'> Arrivals At Each Station <a>"**)** # html line: outputs a link for the user to use if they want to return to the previous webpage.

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<div class='footer'>"**)** # html line: references the footer div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Computer Science: Non-Exam Assessment</h3>"**)** # html line: text within the footer div element is output.

**print(**"<h3>2022/23</h3>"**)** # html line: text within the footer div element is output.

**print(**"</div>"**)** # html line: closes the footer div element defined from the CSS file for structuring the web page.

**print** **(**"</body>"**)** # html line: the closing </body> element for structuring the web page

A screenshot of a computer

Description automatically generated**print** **(**"</html>"**)** # html line: the closing </html> element for structuring the web page

liftdisruptions.pyw:

**import** csv**,** requests**,** json**,** cgi**,** cgitb**,** shutil**,** glob**,** os # Import modules which will be used later in the program.

**import** tubeAPI # Import modules which will be used later in the program.

#form will allow them to input any station AND if they want to see toilets, ramp routes, step free changes

form **=** cgi**.**FieldStorage**()** # Data inputted from the form is stored so that it can be used later in the program.

station **=** form**.**getvalue**(**'station\_lifts'**)** # The station that the user inputted is accessed from the form and stored as the variable called "station".

# Print statements output as html lines:

**print** **(**"Content-type:text/html\n\n"**)** # html line: defines the content type so that the lines are processed and interpretted correctly by the computer.

**print** **(**"<html>"**)** # html line: the <html> element for structuring the web page.

**print** **(**"<head>"**)** # html line: the <head> element for structuring the web page.

**print** **(**"<title>Lift Disruptions</title>"**)** # html line: the <title> element for distinguishing which page the user is on.

**print(**"<link rel='stylesheet' type='text/css' href='/styles.css'>"**)** # html line: references the file used for CSS styling of the webpage (objective 4.2).

**print(**"<meta name='viewport' content='width=device-width, initial-scale=1'>"**)** # html line: formats the webpage so that the content atomatically adjusts its size to fit the screen.

**print** **(**"</head>"**)** # html line: the closing </head> element for structuring the web page.

**print(**"<body>"**)** # html line: the <body> element for structuring the web page.

# Objective 3.1:

**print(**"<div class='navbar'>"**)** # html line: references the navbar div defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<a href='/journeyplanner.html'class='left'>Journey Planner</a>"**)** # html line: the first part of the navbar is for the journey planner - when clicked, the website redirects to the html for the journey planner page.

**print(**"<a href='/map.html'class='center'>Map</a>"**)** # html line: the second part of the navbar is for the map - when clicked, the website redirects to the html for the map page.

**print(**"<a href='/updates.html'class='center'>Status Updates</a>"**)** # html line: the third part of the navbar is for status updates - when clicked, the website redirects to the html for the status updates page.

**print(**"<a href='/arrivaltimes.html'>Arrival Times</a>"**)** # html line: the fourth part of the navbar is for the arrival times - when clicked, the website redirects to the html for the arrival times page.

**print(**"<a href='/liftupdates.html'>Lift Updates</a>"**)** # html line: the fifth part of the navbar is for the lift updates - when clicked, the website redirects to the html for the lift updates page.

**print(**"<a href='/index.html'class='right'>About</a>"**)** # html line: the last part of the navbar is for the information page - when clicked, the website redirects to the html for the about page.

**print(**"</div>"**)** # html line: closing tag for the navbar div defined from the CSS file for structuring the web page.

**print(**"<div class='header'>"**)** # html line: references the header div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Tube Journey Planner</h3>"**)** # html line: text within the header div element is output.

**print(**"<h3>Farheen Fahim</h3>"**)** # html line: text within the header div element is output.

**print(**"</div>"**)** # html line: closes the header div element defined from the CSS file for structuring the web page.

**print(**"<h1>Lift Updates</h1>"**)** # html line: outputs a title for the webpage.

t **=** tubeAPI**.**Tube\_API**()** # Instantiation: create an object called "t" using the class called "Tube\_API" from the "tubeAPI" file which was imported at the start.

ID**,** ICSCODE **=** t**.**get\_station\_id**(**station**)** # Calls the "get\_station\_id" method within the object called "t" with the argument "station", the values returned are stored using the identifiers called "ID" and "ICSCODE.

# Objective 3.3:

endpoint **=** **(**"https://api.tfl.gov.uk/Disruptions/Lifts/v2/?app\_key=61cab23336c147e29c64f5de382d1d91"**)** # the variable called "endpoint" stores the string which is the url for accessing the API information.

data **=** t**.**get\_data**(**endpoint**)** # Calls the "get\_data" method within the object called "t" with the argument "endpoint", the value returned is stored using the identifier called "data".

empty **=** **[]** # Creates an empty list called "empty".

**for** i **in** **range(len(**data**)):** # For loop: iterates through the list called "data".

dictionary **=** data**[**i**]** # Stores the value from the "data" list which has index position i using the identifier called "dictionary".

**if** ID **in** dictionary**[**"stationUniqueId"**]:** # If the station which the user entered is in the data we are iterating through, then it is relevant, and we need to filter for this data, so run the following line of code.

empty**.**append**(**data**[**i**])** # Append the relevent information to the (empty) list so that this list only contains the information on the station the user has entered.

data **=** empty # Stores the list already stored in empty using the identifier "data".

**if** **len(**data**)** **==** 0**:** # If the length of the list called "data" is 0, there is no information on any lift disruptions at this station at the moment, so run the following line of code.

**print(**"<h3> No lift disruptions at this station at the moment :) </h3>"**)** # html line: since there is no information on any lift disruptions at this station at the moment, this line displays a relevant message for the user.

**else:** # If the length of the list called "data" is not 0, then there is information on lift disruptions at the station the user has entered, so run the following lines of code to display the information accessed from the API.

**for** i **in** **range(len(**data**)):** # For loop: iterates through the list called "data" so that all lift disruptions are outputted for the user.

dictionary **=** data**[**i**]** # Stores the value from the "data" list which has index position i using the identifier called "dictionary".

#Creating strings which are ready to be output (involves concatenating strings with strings accessed from the dictionary with information on the lift disruption):

text1 **=** "Message: " **+** **str(**dictionary**[**"message"**])** # Stores a string (which contains a description of the lift disruption) in the variable called "text1".

text2 **=** "Lift Affected (Unique Lift ID): " **+** **str(**dictionary**[**"disruptedLiftUniqueIds"**])** # Stores a string (which contains the lift ID of the lift which is affected by the disruption) in the variable called "text2".

text0 **=** "Lift " **+** **str(**i**+**1**)** # Stores a string (which helps to label each lift disruption when outputting it for the user) in the variable called "text0".

**print(**"<h3> %s</h3>" **%** text0**)** # html line: outputs a line which orders the data by adding a subtitle for which lift disruption it is (e.g. lift disruption 1, 2, 3 etc).

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<h3> %s</h3>" **%** text1**)** # html line: outputs the data from the API which corresponds to the station that the user entered (message describing how long the lift will be affected).

**print(**"<h3> %s</h3>" **%** text2**)** # html line: outputs the data from the API which corresponds to the station that the user entered (which lift is affected).

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<h3> Want to look at other stations' lift disruptions? </h3>"**)** # html line: outputs text on the webpage.

**print** **(**"<a href='../liftupdates.html'> Lift Updates <a>"**)** # html line: outputs a link for the user to use if they want to return to the previous webpage.

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<div class='footer'>"**)** # html line: references the footer div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Computer Science: Non-Exam Assessment</h3>"**)** # html line: text within the footer div element is output.

**print(**"<h3>2022/23</h3>"**)** # html line: text within the footer div element is output.

**print(**"</div>"**)** # html line: closes the footer div element defined from the CSS file for structuring the web page.

**print** **(**"</body>"**)** # html line: the closing </body> element for structuring the web page

Graphical user interface, text, application

Description automatically generated**print** **(**"</html>"**)** # html line: the closing </html> element for structuring the web page

status\_updates.pyw:

**import** cgi**,** cgitb**,** requests # Import modules which will be used later in the program.

**import** tubeAPI # Import modules which will be used later in the program.

form **=** cgi**.**FieldStorage**()** # Data inputted from the form is stored so that it can be used later in the program.

line **=** form**.**getvalue**(**"line"**)** # The line that the user inputted is accessed from the form and stored as the variable called "line".

# Print statements output as html lines:

**print** **(**"Content-type:text/html\n\n"**)** # html line: defines the content type so that the lines are processed and interpretted correctly by the computer.

**print** **(**"<html>"**)** # html line: the <html> element for structuring the web page.

**print** **(**"<head>"**)** # html line: the <head> element for structuring the web page.

**print** **(**"<title>Status Updates for TfL</title>"**)** # html line: the <title> element for distinguishing which page the user is on.

**print(**"<link rel='stylesheet' type='text/css' href='/styles.css'>"**)** # html line: references the file used for CSS styling of the webpage (objective 4.2).

**print(**"<meta name='viewport' content='width=device-width, initial-scale=1'>"**)** # html line: formats the webpage so that the content atomatically adjusts its size to fit the screen.

**print** **(**"</head>"**)** # html line: the closing </head> element for structuring the web page.

**print(**"<body>"**)** # html line: the <body> element for structuring the web page.

# Objective 3.1:

**print(**"<div class='navbar'>"**)** # html line: references the navbar div defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<a href='/journeyplanner.html'class='left'>Journey Planner</a>"**)** # html line: the first part of the navbar is for the journey planner - when clicked, the website redirects to the html for the journey planner page.

**print(**"<a href='/map.html'class='center'>Map</a>"**)** # html line: the second part of the navbar is for the map - when clicked, the website redirects to the html for the map page.

**print(**"<a href='/updates.html'class='center'>Status Updates</a>"**)** # html line: the third part of the navbar is for status updates - when clicked, the website redirects to the html for the status updates page.

**print(**"<a href='/arrivaltimes.html'>Arrival Times</a>"**)** # html line: the fourth part of the navbar is for the arrival times - when clicked, the website redirects to the html for the arrival times page.

**print(**"<a href='/liftupdates.html'>Lift Updates</a>"**)** # html line: the fifth part of the navbar is for the lift updates - when clicked, the website redirects to the html for the lift updates page.

**print(**"<a href='/index.html'class='right'>About</a>"**)** # html line: the last part of the navbar is for the information page - when clicked, the website redirects to the html for the about page.

**print(**"</div>"**)** # html line: closing tag for the navbar div defined from the CSS file for structuring the web page.

**print(**"<div class='header'>"**)** # html line: references the header div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Status Updates</h3>"**)** # html line: text within the header div element is output.

**print(**"<h3>Farheen Fahim</h3>"**)** # html line: text within the header div element is output.

**print(**"</div>"**)** # html line: closes the header div element defined from the CSS file for structuring the web page.

**print(**"<h1>Status Updates</h1>"**)** # html line: outputs a title for the webpage.

# Objective 3.3:

endpoint **=** **(**'https://api.tfl.gov.uk/Line/Mode/tube/Status?app\_key=61cab23336c147e29c64f5de382d1d91'**)** # the variable called "endpoint" stores the string which is the url for accessing the API information.

t **=** tubeAPI**.**Tube\_API**()** # Instantiation: create an object called "t" using the class called "Tube\_API" from the "tubeAPI" file which was imported at the start.

data **=** t**.**get\_data**(**endpoint**)** # Calls the "get\_data" method within the object called "t" with the argument "endpoint", the value returned is stored using the identifier called "data".

info\_list **=** **[]** # Creates an empty list called "info\_list".

line\_disruption **=** **False** # Sets the Boolean value called "line\_disruption" to be False.

**for** i **in** **range** **(len** **(**data**)):** # For loop: iterates through the list called "data".

dictionary **=** data**[**i**]** # Stores the value from the "data" list which has index position i using the identifier called "dictionary".

**if** line **in** dictionary**[**"name"**]:** # Filter for the line that the user has given as input (only runs next lines of code if the line that the user has inputted is in the dictionary)

disruptions **=** dictionary**[**"disruptions"**]** # Stores the value from the dictionary called "dictionary" which corresponds to the disruptions using the identifier called "disruptions"

**if** **len(**disruptions**)** **==** 0**:** # If the list containing the details of disruptions ("disruptions") is empty, then there are no disruptions, so run the following line.

**print(**"<h3> No disruptions </h3>"**)** # html line: tells the user that there are no disruptions by displaying a message.

**else:** # If the list containing the details of disruptions ("disruptions" is not empty, then there are disruptions, so run the following lines of code.

**for** i **in** **range** **(len(**disruptions**)):** # For loop: iterates through the list called "disruptions".

text1 **=** disruptions**[**i**]** # Stores the value from the "disruptions" list which has index position i using the identifier called "text1".

**print(**"<h3> %s </h3>" **%**text1**)** # html line: outputs the data from the API which corresponds to the line that the user entered (if there are any disruptions/ extent of disruptions).

text2 **=** dictionary**[**"lineStatuses"**]** # Stores the value from the dictionary called "dictionary" which corresponds to the lineStatuses using the identifier called "text2".

text2 **=** text2**[**0**]** # Updates the list called "text2" so that it is the first element of that list (which is a dictionary).

text2 **=** text2**[**"statusSeverityDescription"**]** # Updates the dictionary called "text2" so that it becomes the value corresponding to the status severity description in the dictionary.

**print(**"<h3> %s </h3>" **%**text2**)** # html line: outputs the data from the API which corresponds to the line that the user entered (severity description of delays).

# Objective 4.2

**print(**"<style>"**)** # html line: the <style> element for styling the web page.

**print(**".responsive {"**)** # html line: creates a <div> section so that an image can be styled correctly later.

**print(**"max-width: 100%;"**)** # html line: adjusts the maximum width of an image which uses this styling so that it is not scaled up or down (ensures that the image is displayed nicely).

**print(**"height: auto;"**)** # html line: adjusts the height of an image that uses this styling so that it is automatically determined based on the size of the screen/window.

**print(**"</style>"**)** # html line: the closing </style> element for styling the web page.

# Objective 9.2:

# If statements for choosing the image which corresponds to the line that the user has inputted in the form:

**if** line **==** "Bakerloo"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Bakerloo-line-map.png'

**elif** line **==** "Central"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Central-line-map.png'

**elif** line **==** "Circle"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Circle-line-map.png'

**elif** line **==** "District"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/District-line-map.png'

**elif** line **==** "Hammersmith & City"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Hammersmith-city-line-map.png'

**elif** line **==** "Jubilee"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Jubilee-line-map.png'

**elif** line **==** "Metropolitan"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Metropolitan-line-map.png'

**elif** line **==** "Northern"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Northern-line-map.png'

**elif** line **==** "Piccadilly"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Piccadilly-line-map.png'

**elif** line **==** "Victoria"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Victoria-line-map.png'

**elif** line **==** "Waterloo & City"**:**

link **=** 'https://www.london-tube-map.info/wp-content/uploads/2016/04/Waterloo-city-line-map.png'

img\_text **=** "<img src=" **+** link **+** " class='responsive'>" # Stores a string in the variable called "img\_text". THe string is html for outputting an image which is styled by the code above the if statements.

**print(**img\_text**)** # html line: outputs the string stored in the "img\_text" variable in order to output an image.

**print(**"<h3> Want to look at other lines' status updates? </h3>"**)** # html line: outputs text on the webpage.

**print** **(**"<a href='../updates.html'> Status Updates <a>"**)** # html line: outputs a link for the user to use if they want to return to the previous webpage.

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<br>"**)** # html line: <br> tag for line breaks between content

**print(**"<div class='footer'>"**)** # html line: references the footer div element defined from the CSS file for structuring the web page (objective 4.1).

**print(**"<h3>Computer Science: Non-Exam Assessment</h3>"**)** # html line: text within the footer div element is output.

**print(**"<h3>2022/23</h3>"**)** # html line: text within the footer div element is output.

**print(**"</div>"**)** # html line: closes the footer div element defined from the CSS file for structuring the web page.

**print** **(**"</body>"**)** # html line: the closing </body> element for structuring the web page

**print** **(**"</html>"**)** # html line: the closing </html> element for structuring the web page

Graphical user interface

Description automatically generated

tubeAPI.py:

**import** requests**,** json**,** cgi**,** cgitb**,** shutil**,** glob**,** os # Import modules which will be used later in the program.

# Objective 3.3:

**class** **Tube\_API():** # Creates a class for the tube API.

**def** get\_data**(**self**,**url**):** # Creates a method called "get\_data" which takes a single argument which is the endpoint/url the data is being accessed from.

response **=** requests**.**get**(**url**,** timeout**=**10**)** # Uses "get" method in the "requests" module to access/GET the webpage corresponding to the "url" parameter.

**if** response**.**status\_code **>=** 400**:** # If statement: if the status\_code from the previous line is greater than or equal to 400, this means that there was an error, so run the following line.

**raise** **RuntimeError(**f'Request failed: { response**.**text }'**)** # Handling the error: If there is an error, the user will know as this message will be output.

**return** response**.**json**()** # Return the information/data which was accessed from the API as json content.

**def** get\_station\_id**(**self**,**station**):** # Creates a method called "get\_station\_id" which takes a single argument which is the station the user is requesting data on.

endpoint **=** "https://api.tfl.gov.uk/Stoppoint/Search/?query=" **+** station **+** "&modes=tube&app\_key=61cab23336c147e29c64f5de382d1d91" # The variable called "endpoint" stores the string which is the url for accessing the API information.

data **=** self**.**get\_data**(**endpoint**)** # Calls the "get\_data" method within the class with the argument "endpoint", the value returned is stored using the identifier called "data".

info **=** data**[**"matches"**]** # Stores the value corresponding to the "matches" key in the dictionary called "data" using the identifier called "info".

info **=** info**[**0**]** # Stores the first element of the list called “info” using the identifier called “info”.

ID **=** info**[**"id"**]** # Stores the value corresponding to the "id" key in the dictionary called "info" using the identifier called "ID".

ICSCODE **=** info**[**"icsId"**]** # Stores the value corresponding to the "icsID" key in the dictionary called "info" using the identifier called "ICSCODE".

**return** ID**,** ICSCODE # Returns the unique TfL station codes for the station the user has inputted. This is used to identify which station the information is about in the API.