Weather Adventures

Programmers Documentation

Melodie Collins, Joey Le, Peter Nelson, Angela Pelky, Alexa Roskowski - 03/12/2023

1. Introduction

This document serves to explain the aspects of the Weather Adventure program with respect to the different modules and functions, the files associated, and the different use cases.

All code is written in Python and uses the Pillow, Request, and Tkinter libraries.

This documentation assumes a knowledge of python.

2. Source Files

2.1. Activity_Recommender.py

2.1(a) Introduction

Activity_Reccomender.py is part of the Weather Adventures system. This program reads through the activities.json file and based on zoom level and weather data it will spit out recommendations!

2.1(b) sort()

This function sorts a list of recommended activities based on location

Inputs:
- recommended: a list of recommended activities (dictionaries)
- coordinates: a tuple containing two floats

Return:
- top3: a list of top 3 activities (dictionaries) based on location

2.1(c) parse()

This function sorts through the json file and finds activities based on weather and wind parameters

Inputs:

- specific_data: list of activities (dictionaries) from zoom level 1 or 2
- weather: isClear, isRain, isSnoworIce
- wind: Yes or No

Return:

- recommended: a list of recommended activities

2.1(a) zoom()

This function determines which zoom level the user is on and calls parse() and sort()

Inputs:

- zoom: an integer of the current map level
- weather: (3) possible strings "isClear", "isRain", "isSnoworIce"
- wind: (2) possible strings "Yes" or "No"
- coordinates: tuple of floats

Return:

- top3: a list of top 3 activities

2.2. InfoPopup.py

2.2(a) Introduction

This file creates the additional pop-up windows for when a user wants to learn more about a particular area's weather and outdoor activities.

2.2(b) addNewLine()

addNewLines(): a helper function for adding new lines within strings to be displayed on the Tkinter pop-up window; this is used by all helper functions except openWebsite() and createPopUp() listed below.

Inputs:

- longStr: a string that involves the display text potentially considered too long to display on the pop-up window on one line, so it needs to be broken down to multiple lines.
- limitEst: an int to provide a general character limit for each line; this function will add a new line to the first white space after the character found at the limitEst-th character in longStr.

- bulletPoint: a bool to determine if the function is working with outdoor activity-related text which uses bullet points. This function performs an additional step to adjust spacing for text that involves a bullet point.

Return:
- retStr: the string to return that stores the newly formatted string that comes with newlines to allow it to spread across more than one line.

2.2(c) createWeatherDesc()

createWeatherDesc(): a helper function to create the descriptive text to describe the weather on the window pop-up

Inputs:
- WeatherStr: a string that provides the description of the weather provided by the openweathermap API.
- temp: an float that represents the temperature (without considering the unit).
- imperial: a bool to define whether the temperature measurement is in fahrenheit (True) or celsius (False).

Return:
- retStr: a string that displays the exact description for the weather of the window pop-up

2.2(d) createWindDesc()

createWindDesc(): a helper function to create the descriptive text to describe the wind on the window pop-up

Inputs:
- windDir: an int that provides the degree of the wind from 0 to 345 going clockwise
- windSpeec: an int that represents the wind speed without considering the unit
- imperial: a bool to define whether the wind speed measurement is in miles (True) or kilometers (False) per hour

Return:
- retStr: a string that displays the exact description for the wind of the window pop-up

2.2(e) createVisAndHumText()

createVisAndHumText(): a helper function to form the text that describes the weather's visibility and humidity on the pop-up window

Inputs:
- vis: an int representing one's visibility, or the ability to view a certain amount of distance accurately, within an area.

- hum: an int representing the percent of the humidity.
- imperial: a bool to check for a request for the visibility in feet (True) or in kilometers (False).

Return:
- retStr: a string to contain the text to display on the pop-up window regarding the visibility and humidity

2.2(f) openWebsite()

openWebsite(): a button function that would open up the request Outdoor Activity's website when clicked on

Inputs:
- url: a string that represents the website link to access.

Return:
- A webpage opened up in the user's web browser representing the outdoor activity that the button is associated with.

2.2(g) createPopup()

createPopup(): a function that creates a small tkinter pop-up window when a user wishes to know more about a specific region on the grid

Inputs:
- title: a string to display at the top of the pop up as a title
- apiInfo: a list that provides all the weather information collected for area under the specific region.
- imperial: a bool for if the user wants to see the temperature in fahrenheit and wind direction in miles (True) or temperature in celsius and wind direction in kilometers (False).

Return:
- A Tkinter window to view additional weather information and outdoor activities.

2.3. weather.py

2.3(a) Introduction

This file queries openweathermap API with user input and parses response

2.3(b) getWeatherInfo()

This function interacts with the weather api and pulls the weather data into a list

Inputs:

- lat: a float of the latitude
- lon: a float of the longitude
- units: a string of the specific unit

Return:

- infoList: a list of weather information

2.3(c) WeatherSymLookup()

This function initializes the weather description dictionary

Inputs:

- weatherStr: a string of the weather description

Return:

- dict[weatherStr]: a string of the correct image name from weather description

2.4. weather_adventures_main.py

2.4(a) Introduction

This file acts as the intermediary file between the UI and all the all files, as such it imports weather.py, InfoPopup.py, and Activity_Reccomender.py. It uses the os library and the Pillow library.

2.4(b) imageAppender()

This function allows a large image to have a smaller image placed on top of it

Inputs:

- imgLarge: a Pillow image of the background image
- imgSmall: a Pillow image of the image to be place on top
- sizeLarge: a tuple containing the dimensions for the large image (length, width)
- sizeSmall: a tuple containing the dimension for the small image (length, width)

Return: This function will return a Pillow Image where the small image is placed over the large image

2.4(c) image_initializer()

This function initializes all the .pngs for the Tkinter window to display. It uses imageAppender to create the map images with the icons. It calls getWeatherInfo() on each button to store the weather information and figure out which icons to overlay onto the map. All the weather API calls take a bit of time, so there are command line outputs to prove to the user that it is working. This function is called in weather_adventure before the map is displayed.

Inputs: none

Return: This function will store all the initialized images into the directory "images/mapImages_symbols" then either "/Eugene2x2" or "/LaneCounty5x3" depending. The images are labeled #.png if it is the image with the weather icon or #_w.png if it is the image with the wind icon.

2.4(d) more_info()

This function is called in weather_adventure.py when a map button is clicked.

Inputs:

- zoom: this is an integer indicating which map we are looking at (1 being Eugene and 2 being Lane County)
- button_num: an integer that indicates which button has been selected.

Return: By calling zoom() in Activity_Recommender.py and the createPopup() function in InfoPopup.py it will return a popup with more weather information and the top three activities.

2.5. weather_adventures.py

2.5(a) Introduction

weather_adventures.py initializes the user interface with an image window of the Eugene and Lane County maps. When the user opens the application a default map of Eugene will be displayed. The user will have to choose to either zoom in/out of the image, or to click on a specific spot on the grid. Based on these choices, either a new map image will be displayed or a pop-up of activity recommendations and weather data will be displayed. The user should be able to do either of these two actions until they want to close the application.

2.5(b) on_click()

This function is used to remove the current map title

Inputs: none

Return: none

2.5(c) button_click()

This function is for for when a button is clicked; It will change the selected button's image appropriately

Inputs: none

Return: none

2.5(d) zoom()

This function is for the Zoom Buttons; It will change the dimensions of the grid and number displayed

Inputs:

- closer: a bool that determines if the map can zoom in closer
- buttonIn: the zoom in button
- buttonOut: the zoom out button

Return: none

2.5(e) oneToTwo()

This function removes the current buttons for X from the window, but does not destroy them; Next, it adds each button for Y into the Tkinter window

Inputs: none

Return: none

2.5(f) twoToOne()

This function removes the current buttons for X from the window, but does not destroy them. Next, it adds each button for Y into the Tkinter window.

Inputs: none

Return: none

2.5(g) mapTitle()

This function sets the title for the current map

Inputs:

- num: an integer that provides the current map level

Return: none

2.5(h) setEugene()

This function puts all the buttons for viewing Eugene into button1; 2x2

Inputs: none

Return: none

2.5(i) setEugeneEmpty()

This function sets all Eugene button images to Eugene images

Inputs: none

Return: none

2.5(j) setEugeneWeather()

This function sets all Eugene button images to Eugene weather images

Inputs: none

Return: none

2.5(k) setEugeneWind()

This function sets all Eugene button images to Eugene wind images

Inputs: none

Return: none

2.5(l) setLane()

This function puts all the buttons for viewing Lane County into button2; 5x3

Inputs: none

Return: none

2.5(m) setLaneEmpty()

This function sets all Lane County button images to Lane County images

Inputs: none

Return: none

2.5(n) setLaneWeather()

This function sets all Lane County button images to Lane County weather images

Inputs: none

Return: none

2.5(o) setLaneWind()

This function sets all Lane County button images to Lane County wind images

Inputs: none

Return: none

2.5(p) clearWindWeather()

This function sets the wind and weather checkboxes to unselected

Inputs: none

Return: none

2.5(q) clearEmptyWeather()

This function sets the empty and weather checkboxes to unselected
Inputs: none
Return: none

2.5(r) clearEmptyWind()

This function sets the empty and wind checkboxes to unselected

Inputs: none

Return: none

2.5(s) main()

This function sets up the Tkinter Window and runs the program

Inputs: none

Return: none

3. Helper Files

3.1. activities.json

This file stores all the activities that could possibly be recommended to the user.

4. Known Bugs and Errors

We don’t have any safety measures in case of an unexpected weather phenomenon, so we would be unable to recommend any activities in that case.