Eric Duong & Camilo Zapata

Description of the project

The main purpose of this project was to create a Wildfire Emergency Reporting Center for wildfires. Essentially, a user can report a wildfire occurrence in a specific area, the user can provide information about the wildfire such as the environment (densely forested, grass fields, rural area), the size of the fire, and the weather in the area. Depending on the size, environment, and weather conditions of where the wildfire occurs will determine the number of dispatchers that will be dispatched to the scene to put out the wildfire. However, if the fire continues to burn even with the numbers of dispatchers on scene, then more will be requested. Then once the fire is put out, a report will be printed displaying the effects of the fire, the overall growth of the fire, how many dispatchers were on scene when it started and how many it ended, how long it took to put it out and the area of which the fire burned.

Significance of the project

Memories from the Canadian wildfires from last year and news of wildfires from California almost every year, understanding the different types of areas where fire spreads and a proper emergency reporting is important. The significance of the project is to create a tool that allows people to report a fire and based on that report, we can learn how fire spreads, the different types of environments that affect the fire severity and the demonstrate the amount of dispatchers needed to put out the fire. By creating this program, we used Objected Oriented Programming with the use of classes which

also incorporated the concept of inheritance. We used dictionaries, recursion, implemented time, functions, and basic python operations and structures.

Installation and Usage Instructions

A prompt will appear asking the user to enter their name, age, and gender. It will then ask the user various questions regarding the fire such approximate fire size, when inputting size, type the number of acre size following ‘acre’ or ‘acres’. Then it will ask the user the type of weather, when inputting the weather make sure its capitalized and spelled correctly with the 4 options, Windy, Sunny, Rainy, and Snowy. Following that it will ask for the environment where the fire is in with three options, Dense Forest, Grass Fields, and Rural Area. Make sure it's capitalized and spelled correctly. Then a message will be printed saying that help is on the way. Then a live simulation of the wildfire being extinguished is displayed, providing information of the time it takes, the percentage of the fire, as well as the number of dispatchers that are on scene. After the fire is extinguished, a report is created on the fire, displaying all the information about the fire such as the name of the reporter, age and gender. The environment, size, and weather. The time taken, number of dispatchers when the fire started and ended. Then the total amount of area burned from the fire. After the final report the user is then asked if they want to report another fire and if they say ‘yes’ then the algorithm repeats, otherwise the algorithm ends.

Code Structure

Importing time & random

Class Person
Attributes: name, age, gender, environment, size, weather, acre_input=none

def __str__(self):
Method in Person class returns the string representation of the Person class

def PersonReport():
No parameters; this function creates a Person object, provided by the user input values. Includes 3 while loops which ensure valid inputs are used.

Class WildFire(Person)
Inherits the Person classes attributes (name, age, gender, environment, size, weather, acre_input)

Attributes: firePercentage, numDispatchers, totalDispatchers, minutes, areaBurned

def initialDispatchers(self):
Decides how many dispatchers to send initially based on the severity, which is calculated by user input of certain types of environments, weather, and size of the fire.

def CalcSeverity(self):
Calculates the severity of the fire based of the conditions like weather, environment, and size. Returns a number that represents severity

def simulation(self):
Runs _simRecursive() function but updating the previous_fire with firePercentage, basically updating the fire until it is extinguished or equal to 0, so this function provides every new recursive simulation with a different value.

def simRecursive(self, previous_fire):
This function is the backbone of the program. Simulates the fire. The fire can grow while being suppressed. If the firePercentage doesn't die down fast enough (15%>) then more dispatchers will arrive to help extinguish the fire faster by providing more supression (determined by the numDispatchers). Time is also calculated and kept track of. A recursive call is then provided to allow for the simulation of a fire as it is getting extinguished, the base case being firePercentage<=0 (extinguished). When every call is made a live feed of the fire being extinguished at a certain time is printed out, along with the firePercentage and numDispatchers.

def Report(self):
After the fire is extinguished, a final report of the fire is made. Provided is the name of the person, age, and gender. The environment, time taken to extinguish the fire, number of dispatchers when the fire started and ended, and the total area burned.

while True:
This while loop at the end of the code is a part of the testing, while the wildfire.simulation() is running the

user can have a choice to make another report of a wild fire after finishing with one report, simply by typing ‘yes’ to continue.

Functionality and Test Results

```python
def PersonReport():
    #Valid inputs for weather and environment
    valid_weather = ['Windy', 'Sunny', 'Rainy', 'Snowy']
    valid_environment = ['Dense Forest', 'Grass Fields', 'Rural Area']

    print('--- Person Report ---')

    #Input for name, age, and gender of the person reporting the wildfire
    name = input('Enter your name: ')
    age = input('Enter your age: ')
    gender = input('Enter your gender: ')

    #This loop ensures that the user inputs a valid number for the fire size, it also has an exception error for invalid inputs
    while True:
      try:
          size_input = input("Enter approximate fire size (example: '0.5 acre' or '2 acres'): ").lower()
          acre_value = float(size_input.split()[0])
          #Determines the size of the fire based on the inputted number with fires ranging from Small, Medium, to Large
          if acre_value < 1:
              size = 'Small'
          elif 1 <= acre_value <= 3:
              size = 'Medium'
          else:
              size = 'Large'
          break
      #Exception error for invalid inputs
      except(ValueError, IndexError):
          print("Please enter a valid number followed by 'acre' or 'acres' (example: '0.5 acre' or '2 acres').")

    #This loop ensures that the user inputs a valid weather condition
    while True:
        weather = input('Enter current weather (Windy, Sunny, Rainy, Snowy): ')
        if weather in valid_weather:
            break
        else:
            print("Please make sure capitalization is correct or weather inputted is valid.")


    #This loop ensures that the user inputs a valid environment
    while True:
        environment = input('Enter your environment (Dense Forest, Grass Fields, Rural Area): ')
        if environment in valid_environment:
            break
        else:
            print("Please make sure capitalization is correct or environment inputted is valid.")

    #Finally after all the inputs are valid, the Person class is returned with the user's inputs as the parameters
    return Person(name, age, gender, environment, size, weather, acre_input= size_input)
```

```
--- Person Report ---
Enter your name: Eric
Enter your age: 19
Enter your gender: Male
Enter approximate fire size (example: '0.5 acre' or '2 acres'): 5 acres
Enter current weather (Windy, Sunny, Rainy, Snowy): Snowy
Enter your environment (Dense Forest, Grass Fields, Rural Area): Grass Fields
Wildfire reported by Eric
Name: Eric
Age: 19
Gender: Male
Environment: Grass Fields
Size: Large
Weather: Snowy
```

PersonReport() - This function basically returns the users inputs as a Person object. It contains lists of valid inputs (for weather and environments), where then 3 while loops make sure that the input is valid. Acre input has not certain valid input but instead uses the users numerical input and uses if statement inequalities to find the size of the fire (Small < 1, 1 <= Medium <= 3, else Large).

```
def calcSeverity(self):
  #As the user inputs the environment, size, and weather, the severity is determined by the following dictionaries
  environment = {'Dense Forest': 3, 'Grass Fields': 2, 'Rural Area': 1}
  size = {'Large': 3, 'Medium': 2, 'Small': 1}
  weather = {'Windy': 3, 'Sunny': 2, 'Rainy': 1, 'Snowy': 0.5}
  #Returns a number that represents the severity of the fire according to the user's inputs
  return environment[self.environment] + size[self.size] + weather[self.weather]
```

calcSeverity() - This function calculates the severity of the fire by adding the severity values of the users input and returning it, giving a severity of the fire. A dictionary is defined for each attribute with given values.

```
def initialDispatchers(self):
  #Severity of the fire is determined by the environment, size, and weather using the calcSeverity function
  severity = self.calcSeverity()
  #Based on the severity, the number of dispatchers is determined
  if severity >= 8:
    return 10
  elif severity >= 5:
    return 7
  else:
    return 4
```

InitialDispatchers() - This function calculates the number of dispatchers required to put out the fire and is determined by the severity of the fire, which is determined by the calcSeverity() function.

```python
#Recursive function that simulates the wildfire, it ends when the firePercentage is less than or equal to 0, basically when the fire is out
def _simRecursive(self, previous_fire):
  if self.firePercentage <= 0:
      self.Report()
      return

  #Calculates the severity of the fire, the suppression, and the growth of the fire
  severity = self.calcSeverity()
  #Suppresion is calculated by the number of dispatchers and a random number between 1.5 and 3.0
  suppression = self.numDispatchers * random.uniform(1.5, 3.0)
  #Growth is calculated by the severity and a random number between 0.5 and 1.2
  growth = severity * random.uniform(0.5, 1.2)

  #Calculates the new firePercentage by subtracting the suppression and adding the growth to the previous firePercentage
  new_fire = max(0, self.firePercentage - suppression + growth)
  #Drop is the difference between the previous firePercentage and the new firePercentage
  drop = self.firePercentage - new_fire
  self.firePercentage = new_fire
  #Burn rate is calculated by the severity and a random number between 0.1 and 0.4
  burn_rate = severity * random.uniform(0.1, 0.4)
  self.areaBurned += burn_rate
  #Keeping track of the time
  self.minutes += 1

  #If the drop is greater than or equal to 15, the minutes added in a random number between 1 and 4
  if drop >= 15:
    minutes_added = random.randint(1, 4)
  #else if the drop is less than 15 then more dispatchers are sent and the minutes added is a random number between 2 and 6
  else:
    minutes_added = random.randint(2, 6)
    self.numDispatchers += 1
    self.totalDispatchers += 1
    print(f"Fire not decreasing fast enough. Sending more dispatchers: {self.numDispatchers}")

  #Tracks the total time it takes to put out the fire
  self.minutes += minutes_added

  #Printing out time, firePercentage, and number of dispatchers as it goes through the simulation
  print(f"Time: {self.minutes} min | Fire: {self.firePercentage:.2f}% | Dispatchers: {self.numDispatchers}")
  time.sleep(0.5)
  #Recursive call
  self._simRecursive(previous_fire=self.firePercentage)
```

_sumRecursive() - This function is a recursive function, having the base case at the top where the percentage of the fire determines when it stops. Basically, when the fire is at 0% it stops and returns the Report() function. This function is the backbone of the simulation, it uses random number generators to simulate how the fire acts, with the variable's suppression and growth. The new_fire variable along with the firePercentage variable determine the fire being simulated. The drop variable determines the number of

dispatchers that are sent, and the drop is basically how much fire percentage drops. If the fire doesn't get extinguished more than 15% then an additional dispatcher will be sent. Time is also a factor that is being tracked. Finally, the recursion is implemented at the very bottom, the function is continuously called until the fire percentage reaches 0.

```
def Report(self):
  hours = self.minutes // 60
  minutes = self.minutes % 60
  total_burnt = self.areaBurned + float(self.acre_input.split()[0])
  print("\nFinal Report")
  print(f"Reported by: {self.name}, Age: {self.age}, Gender: {self.gender}")
  print(f"Environment: {self.environment}, Size: {self.acre_input}, Weather: {self.weather}")
  print(f"Time Taken: {hours}h {minutes}min")
  print(f"Dispatchers at Start: {self.numDispatchers - (self.numDispatchers - self.initialDispatchers())}")
  print(f"Dispatchers at End: {self.numDispatchers}")
  print(f"Total Area Burned: {total_burnt:.2f} acres")
```

```
Final Report
Reported by: Camilo , Age: 19, Gender: male
Environment: Dense Forest, Size: 2 acres, Weather: Sunny
Time Taken: 0h 36min
Dispatchers at Start: 7
Dispatchers at End: 11
Total Area Burned: 13.48 acres
```

Report() - This function provides a description of the fire after it is extinguished. With the time given, total area burned, as well as the Person attributes provided by the user. Number of dispatchers when the fire started and ended.

Showcasing the Achievement and Project Goals (Verification of Sanity of Code)

```
Welcome to the Wildfire Reporting System!
--- Person Report ---
Enter your name: Camilo
Enter your age: 19
Enter your gender: male
Enter approximate fire size (example: '0.5 acre' or '2 acres'): 2 acres
Enter current weather (Windy, Sunny, Rainy, Snowy): Sunny
Enter your environment (Dense Forest, Grass Fields, Rural Area): Dense Forest
Thank you Camilo , dispatchers are on their way!
```

User inputs information including name, age, gender, fire size, weather, and environment. Then a message including the users name will appear notifying that dispatchers are on the way.

```
Wildfire reported by Camilo  in a Dense Forest with a size of 2 acres and Sunny weather.
Fire not decreasing fast enough. Sending more dispatchers: 8
Time: 5 min | Fire: 93.56% | Dispatchers: 8
Fire not decreasing fast enough. Sending more dispatchers: 9
Time: 12 min | Fire: 81.04% | Dispatchers: 9
Fire not decreasing fast enough. Sending more dispatchers: 10
Time: 19 min | Fire: 70.56% | Dispatchers: 10
Time: 24 min | Fire: 47.02% | Dispatchers: 10
Time: 29 min | Fire: 29.60% | Dispatchers: 10
Time: 32 min | Fire: 14.29% | Dispatchers: 10
Fire not decreasing fast enough. Sending more dispatchers: 11
Time: 36 min | Fire: 0.00% | Dispatchers: 11
```

A report will be displayed with the name of the person who reported it, the environment, size of the fire at which they reported it and the weather conditions. Then a live simulation will appear of the fire being extinguished with the active time, fire percentage and dispatchers.

```
Final Report
Reported by: Camilo , Age: 19, Gender: male
Environment: Dense Forest, Size: 2 acres, Weather: Sunny
Time Taken: 0h 36min
Dispatchers at Start: 7
Dispatchers at End: 11
Total Area Burned: 13.48 acres
```

Once the fire is extinguished a final report will be printed, with all the data collected from the first part, the time it took to extinguish,

the dispatchers at start and at end. And the total Area Burned adds up the initial acre and the amount it burned during the fire.

```
Would you like to report another fire? (yes/no): yes
--- Person Report ---
Enter your name: Eric Duong
Enter your age: 20
Enter your gender: male
Enter approximate fire size (example: '0.5 acre' or '2 acres'): 4 acres
Enter current weather (Windy, Sunny, Rainy, Snowy): Windy
Enter your environment (Dense Forest, Grass Fields, Rural Area): Dense Forest
Thank you Eric Duong, dispatchers are on their way!
```

There will be a prompt at the bottom asking whether you want to report another fire, if user says yes then new report with same questions will be asked.

```
Wildfire reported by Eric Duong in a Dense Forest with a size of 4 acres and Windy weather.
Fire not decreasing fast enough. Sending more dispatchers: 11
Time: 5 min | Fire: 93.37% | Dispatchers: 11
Time: 7 min | Fire: 76.82% | Dispatchers: 11
Time: 10 min | Fire: 57.14% | Dispatchers: 11
Time: 15 min | Fire: 38.93% | Dispatchers: 11
Fire not decreasing fast enough. Sending more dispatchers: 12
Time: 18 min | Fire: 26.19% | Dispatchers: 12
Time: 20 min | Fire: 8.96% | Dispatchers: 12
Fire not decreasing fast enough. Sending more dispatchers: 13
Time: 26 min | Fire: 0.00% | Dispatchers: 13
```

Live simulation will be printed with the time, fire percentage and dispatchers.

```
Final Report
Reported by: Eric Duong, Age: 20, Gender: male
Environment: Dense Forest, Size: 4 acres, Weather: Windy
Time Taken: 0h 26min
Dispatchers at Start: 10
Dispatchers at End: 13
Total Area Burned: 20.45 acres

Would you like to report another fire? (yes/no): no
Thank you for using the Wildfire Reporting System!
```

Then the final report will be printed will the same question asking if you would like to report another fire, if not then the program will end.

Discussion and Conclusions

In this project it comes with some limitations including the options when it comes to size, location, and weather. Having few options, it limits the user's ability to choose a specific condition they would like to choose. Another limitation is case and spelling sensitivity when it comes to inputting the size, location, and weather. Some challenges that we faced was the value of the area burned. If a user were to input a value that was for example, 14 acres, the area burned on the report would come up less than what it started in different trials. We then had to fix it by adding the area burned with the initial input, to get the total area burned. Another challenge was wanting to simulate the fire, using what we learned in this course we figured we had to use recursion so that the fire would get put out until it hit the zero-percentage mark. Other course work that we implemented was classes, object-oriented programming concepts, loops, functions, time, and algorithms. Therefore, concluding this final project of CMPSC 132, we involved the many techniques we learned throughout the semester to put together a well understood program which has innovative/impactful meaning if implemented on a larger more complex scale. Taking the knowledge, we hope to carry it with us further down our educational endeavors to become more professional developers.