```
# Import Section
# Imports the employee, and therefore the person class, for creating instances of all users inputted
from Employee import Employee

#The functions and variables were made by my partner

# Variables Section
firstName = ""  # Declares the first name variable for storing first names inputted
lastName = ""  # Declares the last name variable for storing last names inputted
yearBorn = ""  # Declares the year born variable for storing years of birth inputted
# Declares the password length variable for storing the numerical value of the password length inputted
password_length = ""
# Declares the input correct boolean value for checking if the user input is correct
input_correct = ""
preDefined = False
passwordOptions = ""
all_employee_data_in_tuple = [
]  # Declares a tuple that stores all employee data
username_list = []  # Declares a list that stores all generated usernames
username_list_sorted = [
]  # Decalres a list that stores all generated usernames catgeorized alphabetically
employee_dictionary = {
}  # Declares a dictionary that stores all employee data and their respective usernames and passwords
employee_greeting_and_age_list = [
]  # Declares a list storing the greetings and ages of all inputted users
employee_greeting_age = (
)  # Declares a tuple storing all respective combined greetings and ages of all inputted users
greet = ""  # Declares a variable that stores the greeting for each user
age = ""  # Declares a variable that stores the greeting for each user
# Yes list to allow easier verification
yes_list = ["Yes", "Y", "y", "yes", "YES"]
defPassword = ""


# Functions section
def build_usernames(
    first, last, year, dup
):  # A function that builds usernames based on user input. The 4 parameters are first name, last name, year born, and a
boolean value that checks for duplicates.
    if (
            not dup
    ):  # If no duplicates of the data given to the function are reported, this code block is processed
        username = first[0].lower() + last.lower() + year[
            -2:]  # The username is the lowercase version of the first letter of the first name, the entire last name in
lowercase, and the last 2 digits of the year born
    else:  # if a duplicate is found this block of code runs
        username = first.lower() + last[0:1].lower() + year[
            -2:]  # The username is the entire first name in lowercase, the first letter of the last name in lowercase, and the
last 2 digits of the year born

    return username  # Returns the finished username to the rest of the program
```

```
# Input Section
numOfEmployees = int(input("How many employees do you want to enter? "))
while len(
        all_employee_data_in_tuple
) < numOfEmployees:  # Only runs while the amount of employees is less than 5
    firstName = input("Enter first name: ")  # Asks for the first name
    while len(
            firstName
    ) < 2:  # Repeats this until user enters name greater than 2 characters
        # Asks for the first name
        firstName = input("Enter first name: ")
    firstName = firstName.title()
    lastName = input("Enter last name: ")  # Asks for the last name
    while len(
            lastName
    ) < 2:  # Repeats this until user enters name greater than 2 characters
        lastName = input("Enter last name: ")  # Asks for the last name
    lastName = lastName.title()
    # Asks for the year of birth
    yearBorn = input("Enter your year of birth: ")
    while len(yearBorn) != 4:  # Repeats this until user enters a 4 digit number
        yearBorn = input(
            "Enter your year of birth: ")  # Asks for the year of birth
    preDefined = input("Do you want to set a default password? ")
    if preDefined in yes_list:
        print("We have set the default password for this user. ")
        preDefined = True
        employee_data = [firstName, lastName, yearBorn, 0, False, False]
        all_employee_data_in_tuple.append(employee_data)
        preDefined = False
        continue
    else:
        preDefined = False
        defPassword = ""
        while True:  # Will always occur
            password_length = input(
                "Enter a number between 10 and 16: "
            )  # Asks for a password length between 10 and 16
            try:  # Default procedure
                password_length = int(
                    password_length
                )  # Converts the answer to an integer to compare whether the input is between 10 and 16
                if (
                        password_length >= 10 and password_length <= 16
                ):  # If the number inputted is between 10 and 16, the loop stops
                    break  # The loop breaks and goes to the next statement
                else:  # If the above statement doesn't hold true, this block runs
                    continue  # Else, the loop goes back to the start and asks for the password length
            except:  # If the password cannot be converted to a integer, this block runs
                continue  # The loop goes back to the start and asks for the password length
        special_num_ans = input(
            "Would you like special characters in your password? "
        )  # Asks whether special characters should be in the password
        if special_num_ans in yes_list:  # If the answer is in the yes list, this
```

```
        use_special_chars = True  # This is set to true for the employee function build_password
    else:  # Else, this block runs
        # This is set to false for the employee function build_password
        use_special_chars = False
    special_num_ans = input("Would you like numbers in your password? ")
    # Asks whether numbers should be in the password
    if special_num_ans in yes_list:  # If the answer is in the yes list, this block
        use_nums = True  # This is set to true for the employee function build_password
    else:  # Else, this block runs
        use_nums = False  # This is set to false for the employee function build_password
    print(
        "The first name is " + firstName + ", the last name is " + lastName +
        ", and the year of birth is " + yearBorn + "."
    )  # Shows the user what they inputted for first and last name and the year born
    if use_special_chars:  # If this was declared true, this block runs
        # This will show as a message
        message_special_char = "You want to use special characters in your password."
    else:  # If this was declared true, this block runs
        # This will show as a message
        message_special_char = "You don't want to use special characters."
    if use_nums:  # If this was declared true, this block runs
        # This will show as a message
        message_num = "You want to use numbers in your password."
    else:  # If this was declared true, this block runs
        # This will show as a message
        message_num = "You don't want to use numbers in your password."
    print(
        message_special_char + " " + message_num
    )  # Adds together the 2 messages based on the selection statements above
    input_correct = input("Is this correct? Yes or No "
                          )  # Asks the user whether all input is correct
    if (input_correct in yes_list):  # If all data is correct
        employee_data = [
            firstName, lastName, yearBorn, password_length, use_special_chars,
            use_nums
        ]  # Stores all employee data inputted into a dictionary
        all_employee_data_in_tuple.append(
            employee_data)  # Adds the employee data into a tuple
        firstName = ""  # Resets the first_name variable for the next input
        lastName = ""  # Resets the last_name variable for the next input
        yearBorn = ""  # Resets the yearBorn variable for the next input
        use_special_chars = False  # Resets the use_special_chars variable for the next input
        use_nums = False  # Resets the use_nums for the next input
        input_correct = ""  # Resets the input_correct variable for the next input
    else:  # If the data is not correct
        firstName = ""  # Resets the first_name variable for the next input
        lastName = ""  # Resets the last_name variable for the next input
        yearBorn = ""  # Resets the yearBorn variable for the next input
        use_special_chars = False  # Resets the use_special_chars variable for the next input
        use_nums = False  # Resets the use_nums for the next input
        input_correct = ""  # Resets the input_correct variable for the next input
        continue  # Goes back to the top of the loop
# Process Section
for employee in all_employee_data_in_tuple:  # Processes each employee stored in the tuple
```

```
    employee_first_name = employee[
        0]  # Sets the first_name function variable to the value found in the tuple
    employee_last_name = employee[
        1]  # Sets the last_name function variable to the value found in the tuple
    employee_year_born = employee[
        2]  # Sets the yearBorn function variable to the value found in the tuple
    dup_found = False  # Sets this to false by default

    username = build_usernames(
        employee_first_name, employee_last_name, employee_year_born, dup_found
    )  # Builds a username using the above function with the parameters being the values inputted above

    if username in username_list:  # If a duplicate of info is found
        dup_found = True  # This boolean is set to true
        username = build_usernames(employee_first_name, employee_last_name,
                                   employee_year_born, dup_found)
    username_list.append(username)

    my_employee = Employee(employee[0], employee[1], employee[2], employee[3],
                           employee[4], employee[5])
    if password_length != 0:
        password = my_employee.build_password(False)
    else:
        password = my_employee.build_password(True)

    tuple_temp = (my_employee.greeting(), my_employee.age())
    employee_greeting_and_age_list.append(tuple_temp)

    employee_record = [
        employee_first_name, employee_last_name, employee_year_born, password
    ]
    employee_dictionary[username] = employee_record

username_list_sorted = list(username_list)
username_list_sorted.sort()


# Output Section
for employee_greeting_age in employee_greeting_and_age_list:
    greet = employee_greeting_age[0]
    age = employee_greeting_age[1]
    print(f"{greet}, you are {age} years old.")

if numOfEmployees == 0:
    print("You entered no data!\nHave a nice day!")
else:
    filename = 'employee_data.txt'
    with open(filename, 'w+',) as f:
        for key, value in employee_dictionary.items():
            f.write(str(key) + ": " + str(value) + "\n")
    print(
        f"The employee dictionary has been saved to the file {filename}\nThanks for using the program!"
    )
```

```
#Classes meant to represent a typical person

#This class was made by my partner

import datetime     # Imports the datetime class for the 3rd function

class Person:
    def __init__(self, first_name, last_name, yearBorn):        #Takes in the input of the first name, last name, and year born from the main program
        self.first_name = first_name    #Assigns the value(first_name) of this function from input from the main program
        self.last_name = last_name  #Assigns the value(last_name) of this function from input from the main program
        self.yearBorn = yearBorn    #Assigns the value(yearBorn) of this function from input from the main program
    def greeting(self):     #Takes the object "self" as a parameter
        greeting = f"Welcome, {self.first_name}"        #Finds the value of the first name from the self object and concatenates it with "Welcome"
        return greeting     #Returns the greeting to the main program
    def age(self):      #Takes the object "self" as a parameter
        today = datetime.date.today()    #Finds the current date in full form and assigns it to a variable
        age = today.year - int(self.yearBorn)   #Finds the age by subtracting the current year and the year inputted from the self object

        return age      #Returns the result to the main console
```

```
#Classes meant to represent a typical employee
from Person import Person       #Imports the Person class so that the objects in the Person class also are used for this class
import random       #Imports the random library

class Employee(Person):         #Declares the class Employee extending the class of person
        def __init__(self, first_name, last_name, yearBorn, password_length, use_special_chars, use_nums):  #TMakes itself as an object and takes the first name, last name, year of birth, the numerical length of the password, the use of special characters, and the use of numbers as parameters
                super().__init__(first_name, last_name, yearBorn)       #Extends the class of Person
                self.password_length = password_length      #Makes a variable based on the password_length parameter
                self.use_special_chars = use_special_chars      #Makes a variable based on the use_special_chars boolean parameter
                self.use_nums = use_nums    #Makes a variable based on the use_nums parameter
        def build_password(self, preDefined):
                alphabet = 'AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz'
                special_chars = '!@#$%^&*'
                nums = '0123456789'
                count = 0
                password = ""
                if preDefined == True or self.password_length == 0:
                        password = "2V9PZtfXGf6Ao9"
                else:
                        password = ""
                        while count < self.password_length:
                                randChar = random.randrange(0,52,1)
                                pwChar = alphabet[randChar]
                                password += pwChar
                                pwChar = ""
                                count += 1

                                if self.use_special_chars and count < self.password_length:
                                        randSC = random.randrange(0,7,1)
                                        pwSC = special_chars[randSC]
                                        password += pwSC
                                        pwSC = ""
                                        count += 1
                                if self.use_nums and count < self.password_length:
                                        randNum = random.randrange(0,9,1)
                                        pwNum = nums[randNum]
                                        password += pwNum
                                        pwNum = ""
                                        count += 1
                return password
```