Coursework Coversheet

Module:	Computer Science	Term:	Third
Date:	11 December 2025	Weighting:	
Student First Name:	Darianna		
Student Last name:	Suasti		
Course Tutor:	Claudia Papi		
Class:	Computer Science		

Marks Obtained (out of 100):

Grade:

Declaration that the work submitted is my own. I:

- € declare that this submission is entirely written in my own words and no part has been generated by AI software.
- € declare that this submission has been written with contributions from AI software.

I acknowledge that-

- € used AI for suggestions, generate ideas or understand core concepts as a preparatory activity
- € used AI to write, rephrase, or paraphrase part of the essay
- € used QuillBot, Grammarly or other software to review language

1st marker signature		2nd marker signature	
Date		Date	

At the end of the reference section, list AI tools used and explain how much (in percentage) they have contributed. Failure to acknowledge use of generative AI tools (such as ChatGPT and Bard) is considered a violation of EF's plagiarism standards.

Report project computer science

Problem definition and analysis:

Deciding on appropriate and comfortable clothes every day is a recurring challenge that many people face. A weather application could help select outfits that are appropriate to the forecast. Sometimes, our minds are too busy with other concerns to remember that the clothes you bought years ago are becoming a problem. The system that I designed aims to make it easier for people to pick an outfit.

This project allowed me to practice my abilities acquired during the computer science class. To develop the program, I used a line-by-line approach in Python on Google Colab, implementing programming structures such as functions, conditionals, booleans, and loops. The structure of the program is contained within a function, and the main sequence of steps includes: Log in, Main Menu, and Data Processing. The user must first be asked for the login data. After authentication, the user can visualize the menu, which has three options. The user will then need to enter the temperature, and finally, the program will show the outfit depending on the input entered.

Documented design:

The main goal of this project is to help the user decide what to wear based on the weather. It was created in 4 main phases. The code is inside a function called weather_wardrobe_assistant, which allows the user to access all blocks of code simply by entering the function. Firstly, the Login system asks for a username and password with a three-attempt limit. Second, the loop menu, which allows the user to see the options and insert an input to see a specific section. Third, Section weather, the user types the temperature and weather conditions in an input. Fourth, the suggestion section, where the data of outfits is saved and decides the best clothes based on the weather.

1st marker signature		2nd marker signature	
Date		Date	

I used different types of variables to store the information. For example, I used strings for the username and password, and floating-point numbers (floats) for the temperature to allow decimals. An important part of my project is the dictionary, because to improve the program's intuition, a subclassification was implemented to make the outfit more precise depending on the weather.

General logic:

The program follows a logical step-by-step order. It starts by asking the user to log in. I used a loop to give the user exactly three attempts. If the login is successful, the program enters a while loop to show the Main Menu. When the user chooses Option 1, the program validates the inputs (making sure the temperature is a number). When the user chooses Option 2, the program looks into the dictionary and finds the correct outfit key. Option 3 simply breaks the loop to exit.

Detailed logic:

To begin with the system, a variable was made inside the function. These variables allow us to save data; inside the variables are saved strings, the username and password that will be used to check and allow entry. The second block of variables is to save the data on temperature. Since we do not have that data yet, the variables will not be defined yet. The max_attempts allows us to manipulate conditions for user entry, and finally, the logged_in will be the user's state, defined as false or true.

```
#Define the main function called weather_assistance
#This function contain all the program
def weather_assistance(): # all the comands(login,menu,suggestions) will be exeecuted inside this block
  #create a variables to store the correct data that allow us to check if the user input matches
  valid_username='Darianna06'
  valid_password='darianna2006suasti'

  # Variables to save the future weather data
  temperature= None # the data are not defined yet for the input
  raining_time=None
  windy_time=None

  # This variable defined the maximun numbber of chances the user has to try to log inn
  max_attempts=3

#Boolean variable serve  to save loggin details of the user, Im using a bool because acts like a "flag" a
  #The program can check this value later to determine whether to allow access This improves security and
  logged_in=False #the user has not logged in yet
```

1. Log in system:

The code block begins by taking into account the attempts with a for loop that takes the number of the variable max_attemps, for each attempt the user would be asked for an input the user and password. When the user types the data, the input would be checked with a conditionals and boolean function (if=conditional, and=logical operator, == data identical) to verify the data and allow the access here the status of the user change to true and the loop would be broken.

To close the condition we use else and established the system to maximum attempts where a subtraction was created inside a variable, this was created to save the data of attempts left and established a new condition when the number is reduced to zero and no more attempts can be made to close this condition we print a message to inform the user that access was not permitted and exit the function with return.

1st marker signature		2nd marker signature	
Date		Date	

```
# Create a loop to repeat the block inside the for
for attempt in range(max_attempts): # max attempts=3 opportunities

      #createe an input inside the loop, if the data is not correct the input will ask again
      username = input("Enter your username: ")
      password = input("Enter your password: ")

   #Create a condition to check the data of the input and the store data in variables
            # Boolean and is used to check if both of the input are correct
      if username == valid_username and password == valid_password:
          print(f"Login successful! Welcome, {username}")
          # change the status of the user, now the user could acces to the next code
          logged_in = True
          # the user loged to the account, the loop are not necesary now
          break
  #Close the condition if, when the data are not correct
      else:
        #create a variable and calculate inside the substract of the current attempt number (plus 1 becaus
          attempts_left = max_attempts - (attempt + 1)
          if attempts_left > 0:
              print(f"Incorrect. You have {attempts_left} attempts left.")
          #close the condition if the attempts end
          else:
            print('You have exceeded the number of attempts, Access denied')
#Stablish a condition if the log in was not succesfull.
if not logged_in:
  return  # Exit function if login fails
```

Programing Tools:

- ➢ Variables and Assignment: Storing and naming data (like text or numbers) for use throughout the code
- ➢ For loop : Repeating a specific block of code a set number of times, primarily used here to control the maximum login attempts.
- ➢ Break: Immediately stops the execution of the innermost loop (used when login is successful)
- ➢ Function range(): Generates the sequence of numbers that dictates how many times the for loop will run.
- ➢ Boolean Flags: A variable (logged_in) that holds a True or False value to track the state of the program
- ➢ Conditional Structures such as if, elif serves to execute different sections of code based on whether a specified condition is met
- ➢ Logical Operators (and): Used to combine multiple conditions, requiring that all parts are true for the combined condition to pass

1st marker signature		2nd marker signature	
Date		Date	

- ➢ return: Stops the execution of the entire function
- ➢ Input/Output Functions: Using input() to gather data from the user and print() to display messages

2. Menu architecture:

```
#Create the principal menu, While true is a infinite comand
while True:
  print('Principal menu:')
  print('1. Enter todays weather')
  print('2. Get clothing suggestion')
  print('3. Exit')
  #Ask for input to "enter" the choice in the menu
  choice=input('Enter your choice 1-3: ')

#Establish the pourpose for each choice in menu
    if choice =='1':
```

If the data log in status changes to true, the user could see an infinite menu, where the user could enter an input which will be called choice a variable and this will serve to separate the different parts within the program. Infinite menu is created with a while and a truth , this allows the menu to continue running.

Programing tools:

- ➢ while Loop is a control flow statement that repeats a block of code as long as its condition remains true.
- ➢ Infinite Loop (while True): A specific use of the while loop to create a structure that runs continuously until explicitly stopped (used here for the Main Menu).
- ➢ String Comparison: Using the equality operator (==) to check if the user's input string matches a menu option string

3. Weather architecture:

In the first menu option, the system will request input to collect data necessary for suggesting an outfit. The first input is the temperature in Celsius, expected as a float number.Additionally, the input is converted to lowercase to prevent errors. A system was created where if the user answers unexpectedly, the program will continue asking for a yes or no

string. This was achieved using a while loop inside the variable where the answer to the "yes" or "no" question should be stored. If the answer is not as expected, a message will be printed to the user. This system works for the questions of whether it is raining and whether it is windy.

```
#Establish the pourpose for each choice in menu
    if choice =='1':

    #Save the input into the variables
        #Ask the user for the temperature, specify the characters expected
      temperature=float(input('Introduce the temperature in Celsius: '))

      #Ask the user for an input  filter:rainy
      raining_time=str(input('It is raining? Yes or No: '))
      # transform the input into low cases
      raining_time=raining_time.lower() #another way of use the function .lower()

    #Create a loop to ask each time that the input do not be yes or no
      #Stablish a rule, answer allowed yes or no
      while raining_time not in ['yes','no']:
        raining_time=str(input('It is raining? Yes or No: ').lower())
        #Condition if the input keeps entering incorrectly
        if raining_time not in ['yes','no']:
          print('Anwser not accepted Please enter Yes or No: ')

    #Ask the user for an input filter:windy
      windy_time=str(input('It is windy? Yes or No: '))
      #Use the function lower to tansform the input
      windy_time=windy_time.lower()

      #Loop to ask each time the nput do not be yes or no
      while windy_time not in ['yes','no']:
        #Ask again, write again the variable ad the input
        windy_time=str(input('It is windy? Yes or No: ').lower())
        #condition if the input keeps entering incorrecly
        if windy_time not in ['yes', 'no']:
```

Programing tools

- (float) serves to convert data from one type (like a string from input()) into another data type (like a floating-point number or number with decimals) to allow for mathematical operations.
- String (.lower()) built-in function that belongs to a string object, used here to convert all characters in the input to lowercase for uniform validation.

1st marker signature		2nd marker signature	
Date		Date	

- ➢ Data Validation with while loop, that continues to prompt the user until the input matches an expected set of values.
- ➢ Lists [ ] is a data type in Python used here to define the set of objects.
- ➢ (not in) is logical operator that checks if an item is not present within a sequence

4. Suggestion Architecture:

This is the choice number two in the menu, were the suggestion are saved in dictionary's and one subcategory was created for an best personalized outfit, inside this function was created the conditions dependent the input of the weather and also a condition if the variable of audit still empty this category cannot be running

```
  elif choice=='2':
#Store the data within libraries, this allows us to classify several objects within a category
    outfit_suggestions = {
    #Inside the dictionary we will open another dictionary so we will have a category between another, more keys.
        "Freezing": { # Less than -3°C
                  "dry": " thermal layers, a turtleneck, a fur coat, flared jeans, and platform booties ",
                  "rain": " thermal layers, a turtleneck, a fur coat UNDER a waterproof trenchcoat, and Biker Boots "
              },
        "Very_cold": { # -3°C to 0°C
                  "dry": " comfy hoodie, a bomber jacket, 'snazzy jeans', and Converse",
                  "rain": " comfy hoodie, a leather jacket, 'snazzy jeans', and Biker Boots (better for wet ground)"
              },
        "Cold": { # 1°C to 4°C
                  "dry": " cable-knit sweater, a miniskirt with cool tights, and Biker Boots ",
                  "rain": " turtleneck, a leather jacket, a plaid miniskirt with tights, and platform booties"
              },
        "Cool": { # 5°C to 14°C
                  "dry": " Henley top or off-shoulder sweater, flared jeans, and classic loafers (Old Money/Scandi vibes)"
                  "rain": " Henley top, a trenchcoat, straight leg jeans, and classic loafers"
              },
        "Warm": { # 15°C to 25°C
                  "dry": " cute tee or polo shirt, wide leg jeans, and classic sneakers. (Y2K/Star Girl vibes)",
                  "rain": " lace border tank top, a zip-up hoodie, wide leg jeans, and platform sneakers"
              },
        "Hot": { # More than 25°C
                  "dry": " lace cami or ruffle tank top, a miniskirt, and platform sandals or leather slides",
                  "rain": " 3 button top or square neck top, shorts, and leather slides (Bring an umbrella!)"
              }
          }
  #Verify with a condition, If the variable of temperature it has been completed
    if temperature is None:
```

1st marker signature		2nd marker signature	
Date		Date	

Programming tools:

- Infinite Loop: Keeps a section of code (like a menu) running continuously until a break is forced While true
- Type (float, string): convert data from one type to another for accurate process
- String lower case (.lower()) created for text manipulation, used to standardize input to lowercase for reliable validation.
- (in / not in): Checks if a specific value is present or absent within a collection of data
- Lists []: Ordered collections of items, used here to define the set of acceptable valid inputs.
- Dictionaries {}: Data structures that store information in key-value pairs, ideal for categorizing and looking up suggestions
- Using one dictionary as the value inside another to create multi-level data classifications
- is None, verifies if a variable has been assigned a value yet, ensuring necessary data has been collected before proceeding
- Ellif conditions: Allows for checking multiple, mutually exclusive conditions sequentially, used to determine temperature ranges.
- Compound logical Operators, combining multiple comparisons in a single expression to define a value range -3 <= temperature <=

Challenges:

The core purpose of the weather wardrobe system, suggesting clothes based on weather, was successfully achieved with some spelling errors since I forgot details, for example, the index of conditions and loops, as well as the spelling of some variables. However, during testing, a critical weakness was identified regarding system stability. When I tried in the section for entering weather data, I tested it and accidentally entered the answer in

capital letters; the program stopped running. On another occasion, I didn't type correctly. I found it necessary and practical to design a system that would continue asking for input until the expected answer was received without having to restart the session. The initial programming flaw was a lack of a mechanism to force the user to re-enter data if their input did not match the expected valid options, for example, enter maybe instead of “yes” or “no”. This caused the program to either execute logic incorrectly or crash, undermining the system's reliability. The solution to address this was enhanced by implementing a loop using the while statement combined with the operator (not in).

This structure ensures that the program does not proceed until valid data is provided:

1. The Loop Condition**,** while loop is established, continues running as long as the user's input (raining_time, for example) is not in the predefined list of accepted answers (e.g., ['yes', 'no']).

Code in the project: while raining_time not in ['yes', 'no']:

2. Inside the loop, the user is repeatedly asked for the input.
3. The .lower() string method is applied to the user's input to standardize the answer. This ensures the comparison with the valid list is accurate regardless of capitalization.

Possible improvements:

I really would like to improve this project, I would explore creating styles that adapt to the user's mood to enhance personalization. I was thinking of including another question: 'How do you feel today?'. Depending on that answer and the weather, the results could be much more personalized. To make this work, I would create a 'Feelings' option in the main menu. We would need to update our dictionary structure by placing specific mood keys inside the weather keys. This links the outfit suggestion to both

1st marker signature		2nd marker signature	
Date		Date	

the climate and the user's emotion. Finally, I would implement this as an optional function using a conditional block, this way, the program still runs perfectly even if the user doesn't want to enter a feeling. For example:

weather_outfits = {

"rainy": { "happy": "Colorful raincoat", "sad": "Cozy grey sweater"

} }

```
weather_outfits = {
    "rainy": {
        "default": "Waterproof jacket and boots",
        "happy": "Colorful raincoat",
        "sad": "Cozy grey sweater"
    }
}

def suggest_outfit(weather, mood=None):
    moods = weather_outfits.get(weather, {})
    return moods.get(mood, moods.get("default", "Basic outfit"))
```

To enhance the outfit suggestions, this part of the code uses a nested dictionary structure that pairs weather conditions with user moods, allowing for customizable recommendations if the user feels happy. The function suggest_outfit is designed for flexibility and is used to join mood-based matches and weather but creates a safety net by falling back to a default option if the mood input is missing or the weather is not recognized. This demonstrates how nested data and conditional logic can be combined to build a system that adapts to the user's needs without causing errors. This project has helped me develop my skills and my own interest in learning about coding, and it has shown me how details can

improve the architecture of a code. This project has allowed me to develop my programming skills and has further sparked my interest in coding. It has shown me in a practical way how attention to details would

1st marker signature		2nd marker signature	
Date		Date	