HW 7 · PYTHON 3 · Individual

Introduction

Assignment Goals

The purpose of this assignment is to learn how to implement loops in Python.

Successful Completion

This assignment has two tasks.

1. Read *Notes Before You Start*, on **Page 1**.
2. Read each task carefully. You are responsible for following all instructions within each task. The deliverables list within each task contains everything you are expected to submit.
3. When your work is complete, confirm your deliverables are submitted to Gradescope.
4. Late submissions will be accepted up to 24 hours after the due date. There is a 25% late penalty.

Notes Before You Start

Gradescope

You will submit all your deliverables to Gradescope for grading. View the Gradescope help for online assignments if you need assistance with submitting your work.

Accessing Gradescope for the first time in ENGR 133?

1. Log into Brightspace and open your ENGR 133 course.
2. Click **Content** from the black menu ribbon at the top of the page.
3. Click **Welcome to ENGR 133** from the Table of Contents in the left sidebar.
4. Click the link titled **Click here to access Gradescope**.
5. Select the assignment you are ready to submit.

Opening Gradescope through Brightspace will auto-enroll you in the Gradescope course for your section.

You can access Gradescope through Brightspace throughout the semester.

Guidelines for Individual Tasks

- You may seek help on individual tasks from classmates, the instructional team, or others but the work that you submit should always be your own.
- If you collaborate with others and use information developed together or by someone else, ALWAYS document and reference that material.

Task #6

Objectives

Predict the output of a complete for and while loop in Python using flowcharts; Use loops to conduct repetitive operations to perform numerical operations in Python; Create, manipulate, and read from arrays, lists, and dictionaries in Python; Manipulate and extract information from lists, arrays, and dictionaries in Python; Use loops to compare and evaluate associated elements in data sets in Python.

Background

Bioinformatics is an interdisciplinary field that develops and applies computational methods to analyze large collections of biological data, such as genetic sequences, protein samples, and cell populations.

In this assignment, you are asked to create a Python program that maps different chromosome combinations to their corresponding phenotypes, a key concept in genetics. The program takes user input and delivers output based on the genetic concept of dominant and recessive traits, which is a common task in bioinformatics. These kinds of programs are essential in analyzing and interpreting biological data, especially genetic data, which is a major part of the bioinformatics field.

Chromosomes carry the genetic instructions used in the growth, development, functioning, and reproduction of all living organisms. Different combinations of chromosomes can lead to different observable physical attributes, known as phenotypes. In this assignment, you will write a Python script that maps the chromosome combinations to their respective phenotypes using the provided flowchart.

Flowcharts are visual representations of processes or workflows which help in planning, developing, and communicating their ideas more clearly. They are critical tools in computer programming, as they help in understanding the flow of control and data in a program.

Submissions

Gradescope Assignment	HW 7 – PY 3 Individual
Task Type	Individual
Deliverables	☐ `Py3_Task6_username.py`

Task Instructions

Review the flowchart provided in Figure 1.

Translate the flowchart into Python code. Your script should include the following:

a. Import a list of features available for chromosome-to-phenotype mapping using the ***list_of_features.csv***
    a. *Hint: use the NumPy function* genfromtxt() *to create an array from the csv file*
b. Print the available features that the user can select.
c. Prompt the user to select a feature they are interested in.
    a. If the user does not enter a valid feature, print an error message.
d. After the user has selected a feature, print the genotypes for the feature (e.g. BB, Bb, or bb).
e. Prompt the user to select one of the genotypes.
    a. If the user does not enter a valid feature, print an error message.

f. Once the user has input the genotype, print the corresponding physical attribute. For example, if the feature is eyebrows and the chromosome type is BB or Bb, output "Thick Eyebrows". If chromosome type is bb, output "Thin Eyebrows".
g. After the physical attribute is displayed, ask the user if they want to run the program again. End or continue the program according to the input from the user.
h. Save your file in a Python file called: `Py3_Task6_username.py`

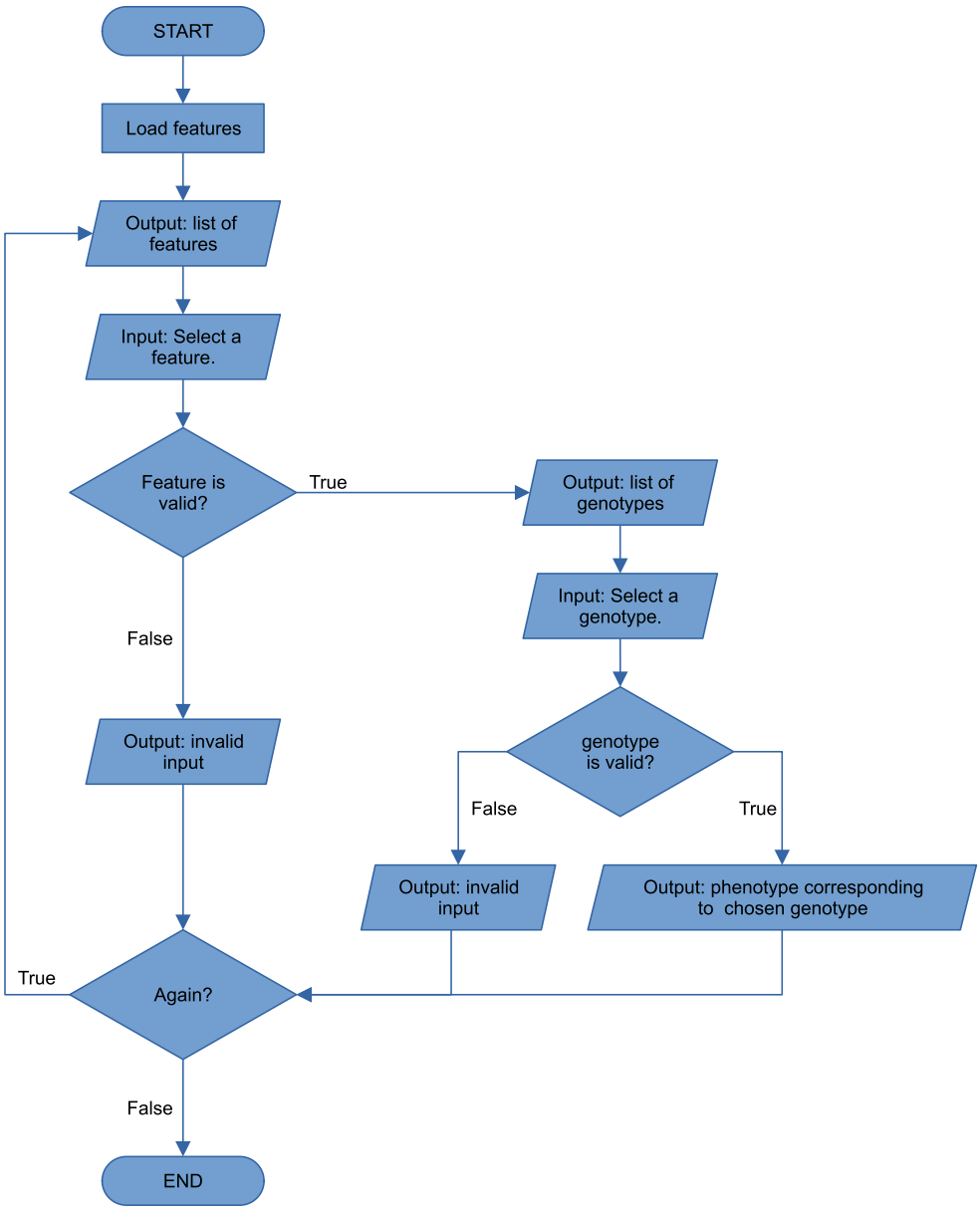

Figure 1: Flowchart for Task 6, which converts chromosome types to phenotypes.

Sample Input and Output:

This example shows the expected format of the output (inputs bold) when the program is run. Please note: the list of AVAILABLE FEATURES is long, so for this sample much of that list has been replaced with "…". Your program should print out the entire list of available features every time.

```
AVAILABLE FEATURES:
Head
Chin
...
Hair color
Hair type
Please select a feature: Chin
POSSIBLE GENOTYPES:
CC
Cc
cc
Please input the genotype: CC
This corresponds to the physical attribute: Square Chin
Would you like to run again? (y or n): y
AVAILABLE FEATURES:
Head
Chin
...
Hair color
Hair type
Please select a feature: Head
POSSIBLE GENOTYPES:
SS
Ss
ss
Please input the genotype: CC
Invalid input.
Would you like to run again? (y or n): y
AVAILABLE FEATURES:
Head
Chin
...
Hair color
Hair type
Please select a feature: Dog
Invalid input.
Would you like to run again? (y or n): y
AVAILABLE FEATURES:
Head
Chin
...
Hair color
Hair type
Please select a feature: Hair type
POSSIBLE GENOTYPES:
AA
Aa
aa
Please input the genotype: aa
This corresponds to the physical attribute: Curly Hair
Would you like to run again? (y or n): n
```

Task #7

Objectives
Use for loops to iterate through arrays. Perform operations with NumPy arrays. Change the shape of arrays.

Background
Sometimes when working with data you will be interested in intermediate values that are between the known data points. Since those values were not collected when the dataset was created, they will have to be interpolated using an appropriate model. Data interpolation is a useful tool in digital signal processing, image processing, weather modeling, and other fields.

In this problem you are given a dataset that contains hourly weather data collected from a station in Bedford, Indiana in 2022. We previously looked at a slice of this dataset in the EX 3 module. This dataset and a description of the data it contains can be found at https://www.ncei.noaa.gov/pub/data/uscrn/products/hourly02/. In this task you will be working with just the date, time, and maximum and minimum temperatures. Some of temperature values are missing from the dataset. Any missing values are given a placeholder value of -9999.

Submissions

Gradescope Assignment	HW 7 – PY 3 Individual
Task Type	Individual
Deliverables	☐ `Py3_Ind_`*`username`*`.pdf` ☐ `Py3_Task7_`*`username`*`.py` ☐ `Py3_Task7_dataset_`*`username`*`.csv`

Task Instructions
In this task you will write a program that imports a dataset, calculates new values from the data, interpolates values from the data, and then exports the new dataset. Follow the steps below. Read the instructions, then draft a flowchart for this task and save it as PY3_Ind_flowchart_*username*.pdf

This task has its own template file that contains two functions in addition to the `main` function.

1. `load_data()`
    a. Opens the data from the text file CRNH0203-2022-IN_Bedford_5_WNW.txt
    b. Returns an 8760 x 4 NumPy array
2. `export_data(array)`
    a. Accepts a NumPy array as an argument
    b. Saves the NumPy array as a csv in the working directory.

Part A: Cleaning the Data

1. Download the data file (CRNH0203-2022-IN_Bedford_5_WNW.txt) and the template file (py3_task7_template.py) and fill out the header information.
    a. You may open the data file and review the information it contains. The data file does not contain any headers. You can read more about what each column is in the link

provided in the **background** section. The python template file will import the following columns: date (YYYYMMDD), time (hours), maximum temperature (Celsius), minimum temperature (Celsius).

b.

2. In your main program, create an array using the provided `load_data` function.
3. Use at least one for loop to iterate through both temperature columns and replace any values that are -9999 according to these specifications:
    a. If both values before and after a -9999 are not -9999, replace the -9999 with the average of the values before and after -9999.
    b. If only one of the values before or after a -9999 is not -9999, replace the -9999 with that value.
        i. *Hint: This can be achieved by using the xor function in a conditional statement*
    c. If both values before and after a -9999 are also -9999, replace the -9999 with the nearest non-9999 value from a previous row.

Part B: Expanding the dataset and interpolating data

Now you will expand the dataset so that it has data for every half hour instead of every hour. To do this you will insert rows between the existing rows and then fill in the appropriate data.

4. Use the numpy.zeros function to create an array that has dimensions 16919 x 4.
5. Insert the data in your array from part A into the even numbered rows (starting with 0) of the array you created in step 4.
    a. *Hint: You can use either a loop or array slicing to achieve this. Recall that when slicing an array you can define the start, stop, and step values.*
6. Fill in the blank, odd numbered rows as follows:
    a. Fill in the date information with the appropriate date.
    b. Fill in the time information with half hour times between the original times (e.g., 130, 230, 330, etc.)
    c. For each temperature column, fill in the empty rows with the mean of the rows directly above and below.
7. Use well formatted print statements to print to the screen the maximum, minimum, mean, median, and standard deviation of the mean temperature for the whole year. Be sure to display an appropriate number of decimal places and include units in your print statement.

Submit your program, flowchart, and the csv output to Gradescope.