BSc (Hons) Artificial Intelligence and Data Science

Module: CM1601

Programming Fundamentals

Report for Python Courseworks

Module Leader: Ms. Sachinthani Perera

RGU Student ID : 2330943

IIT Student ID : 20231421

Student Name : Dilakshan Rahul Surendrabose

Contents

Introduction

The following report presents a rough overview for a text based simulation for a horse race event named ‘Rapid run’ created using python. This report will cover the functionalities used, include descriptions of the main functionalities used and portray flowcharts for the visual representation of how the game would function.

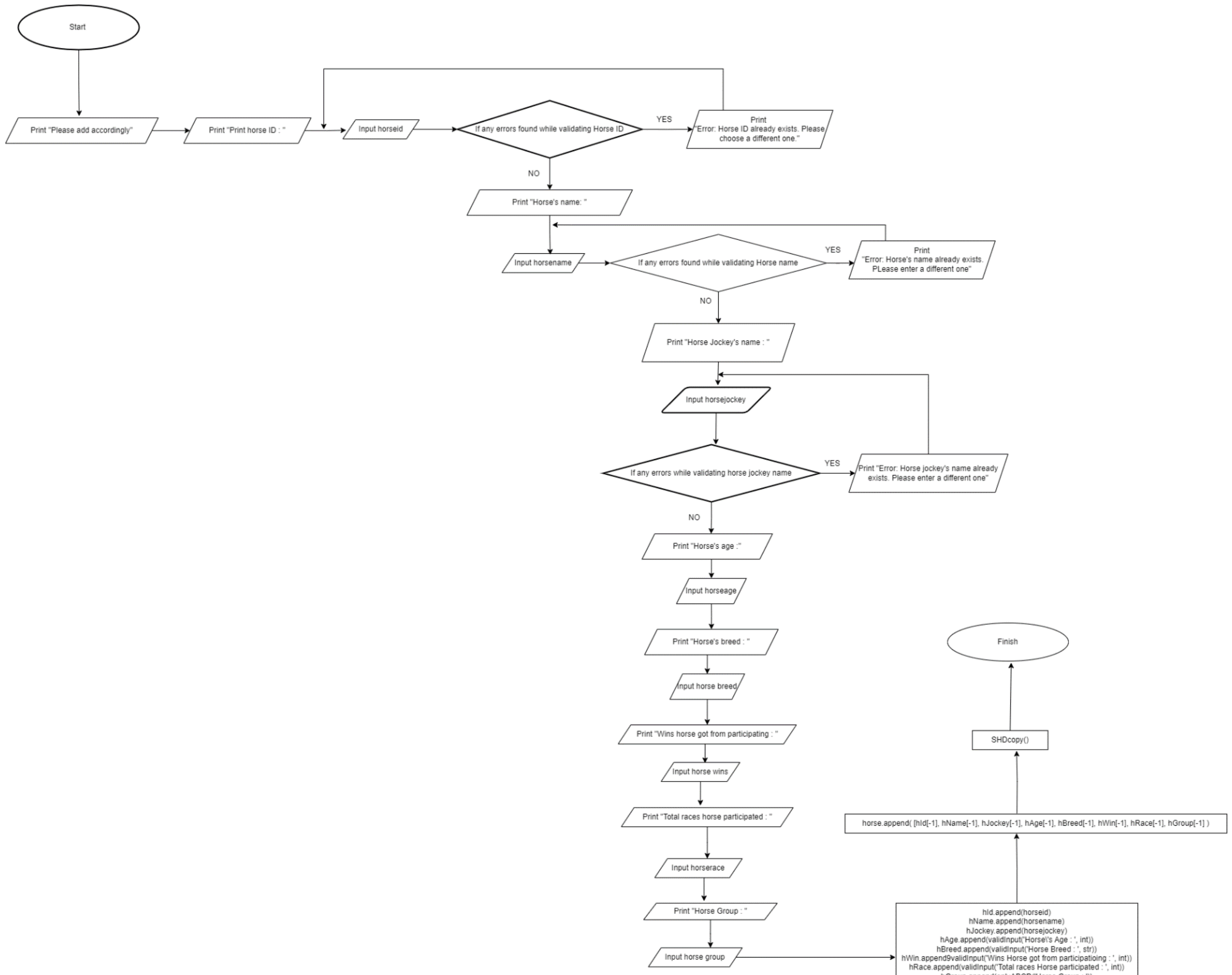
Assumptions/Notes:

horse ID, horse name, horse jockey must be unique when inputing

SHDcopy() save in file called Horse_Detail.txt which stores the details as temporary file
Where SHD() save in the main file Horse.txt

Flowcharts

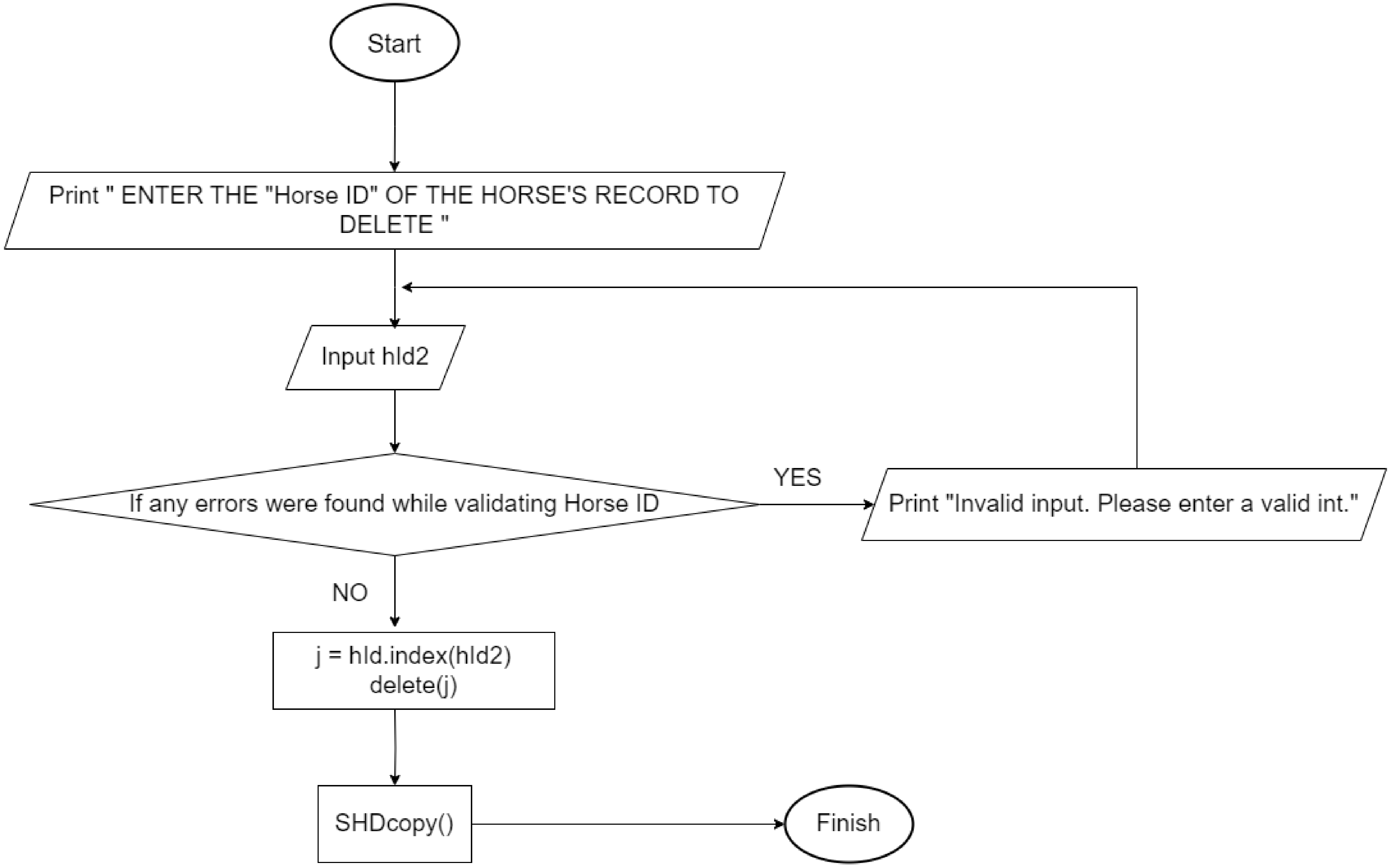
AHD() / Add Horse Details


Start
Print "Please add accordingly"
Print "Print horse ID : "
Input horseid
If any errors found while validating Horse ID
YES
Print "Error: Horse ID already exists. Please choose a different one."
NO
Print "Horse's name: "
Input horsename
If any errors found while validating Horse name
YES
Print "Error: Horse's name already exists. PLease enter a different one"
NO
Print "Horse Jockey's name : "
Input horsejockey
If any errors while validating horse jockey name
YES
Print "Error: Horse jockey's name already exists. Please enter a different one"
NO
Print "Horse's age :"
Input horseage
Print "Horse's breed : "
Input horse breed
Print "Wins horse got from participating : "
Input horse wins
Print "Total races horse participated : "
Input horserace
Print "Horse Group : "
Input horse group
Finish
SHDcopy()

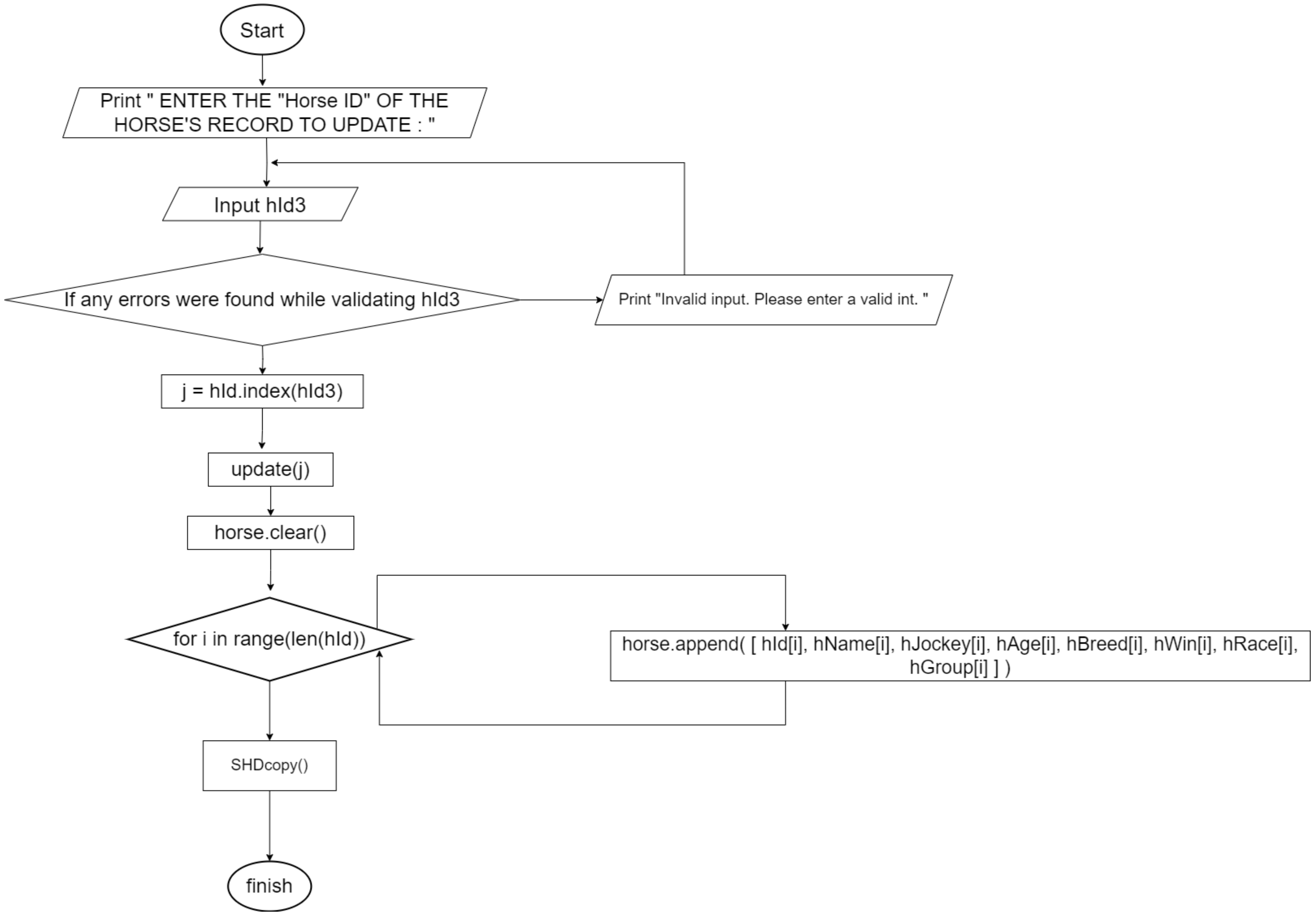
```
horse.append( [hId[-1], hName[-1], hJockey[-1], hAge[-1], hBreed[-1], hWin[-1], hRace[-1], hGroup[-1] )
```

```
hId.append(horseid)
hName.append(horsename)
hJockey.append(horsejockey)
hAge.append(validInput('Horse\'s Age : ', int))
hBreed.append(validInput('Horse Breed : ', str))
hWin.append9validInput('Wins Horse got from participatioing : ', int))
hRace.append(validInput('Total races Horse participated : ', int))
```

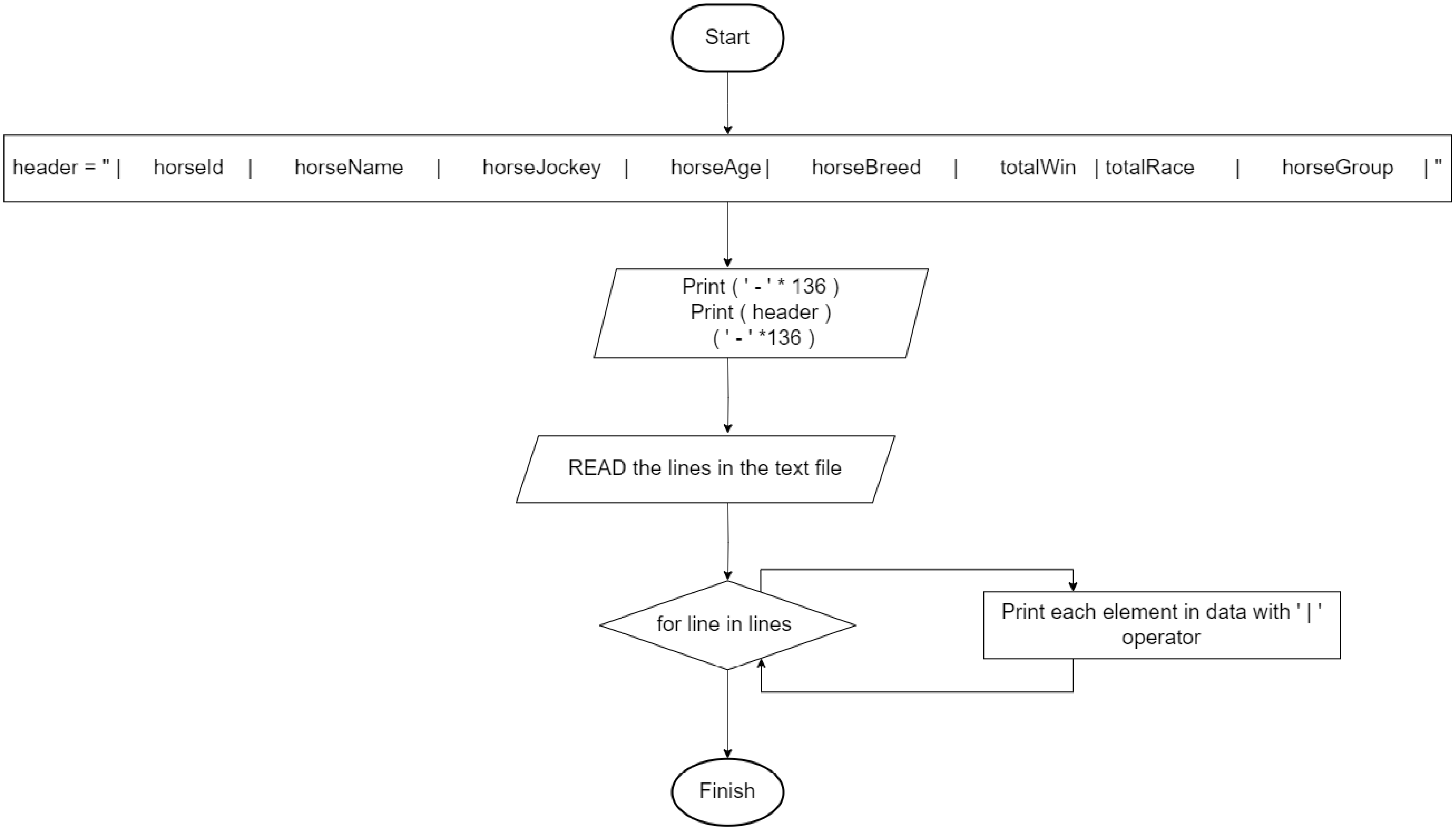
DHD() / Delete Horse Details


Start
Print " ENTER THE "Horse ID" OF THE HORSE'S RECORD TO DELETE "
Input hId2
If any errors were found while validating Horse ID
YES
Print "Invalid input. Please enter a valid int."
NO
j = hId.index(hId2)
delete(j)
SHDcopy()
Finish

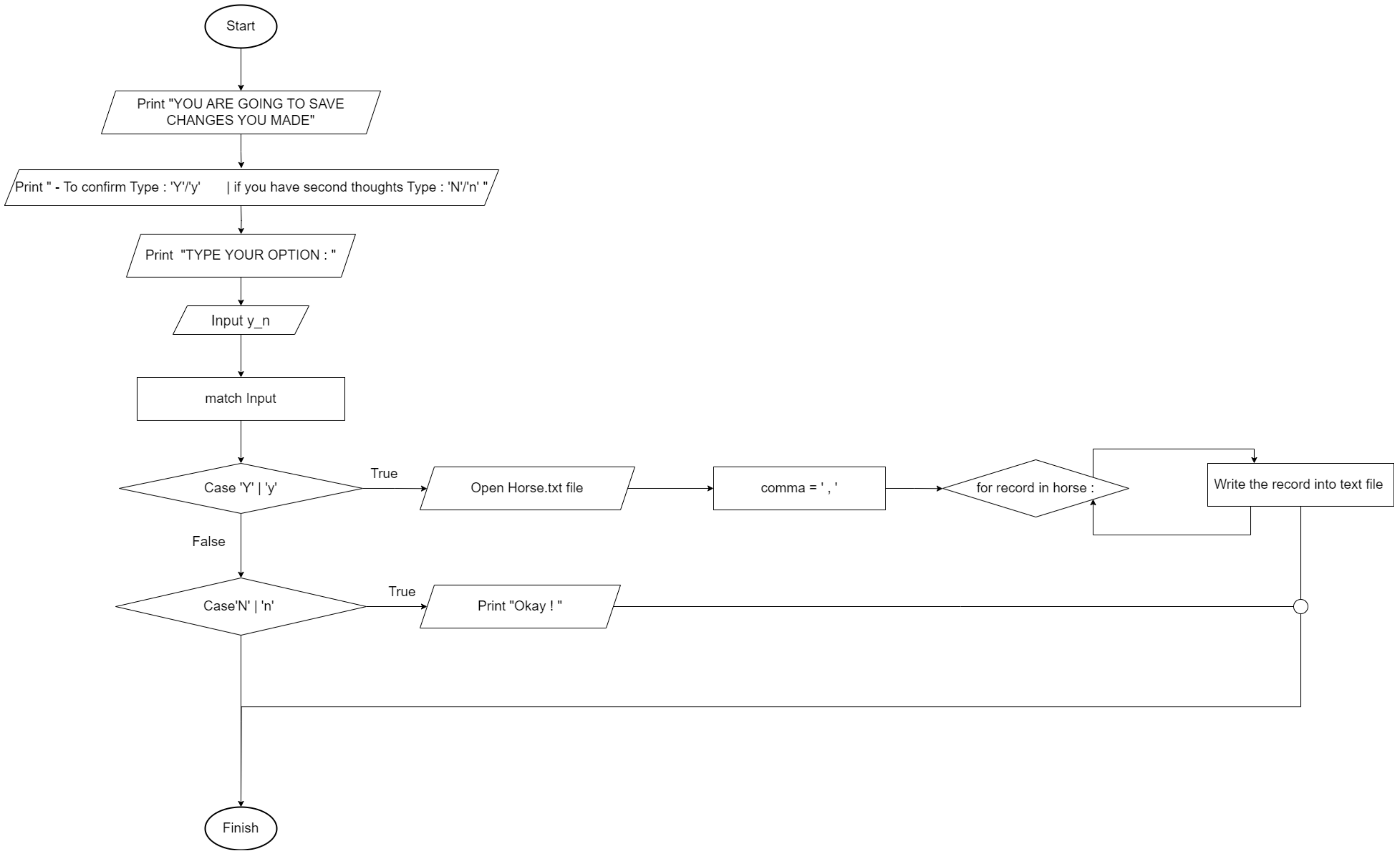
UHD() / Update Horse Details


Start
Print " ENTER THE "Horse ID" OF THE HORSE'S RECORD TO UPDATE : "
Input hId3
If any errors were found while validating hId3
Print "Invalid input. Please enter a valid int. "
j = hId.index(hId3)
update(j)
horse.clear()
for i in range(len(hId))
horse.append([hId[i], hName[i], hJockey[i], hAge[i], hBreed[i], hWin[i], hRace[i], hGroup[i]])
SHDcopy()
finish

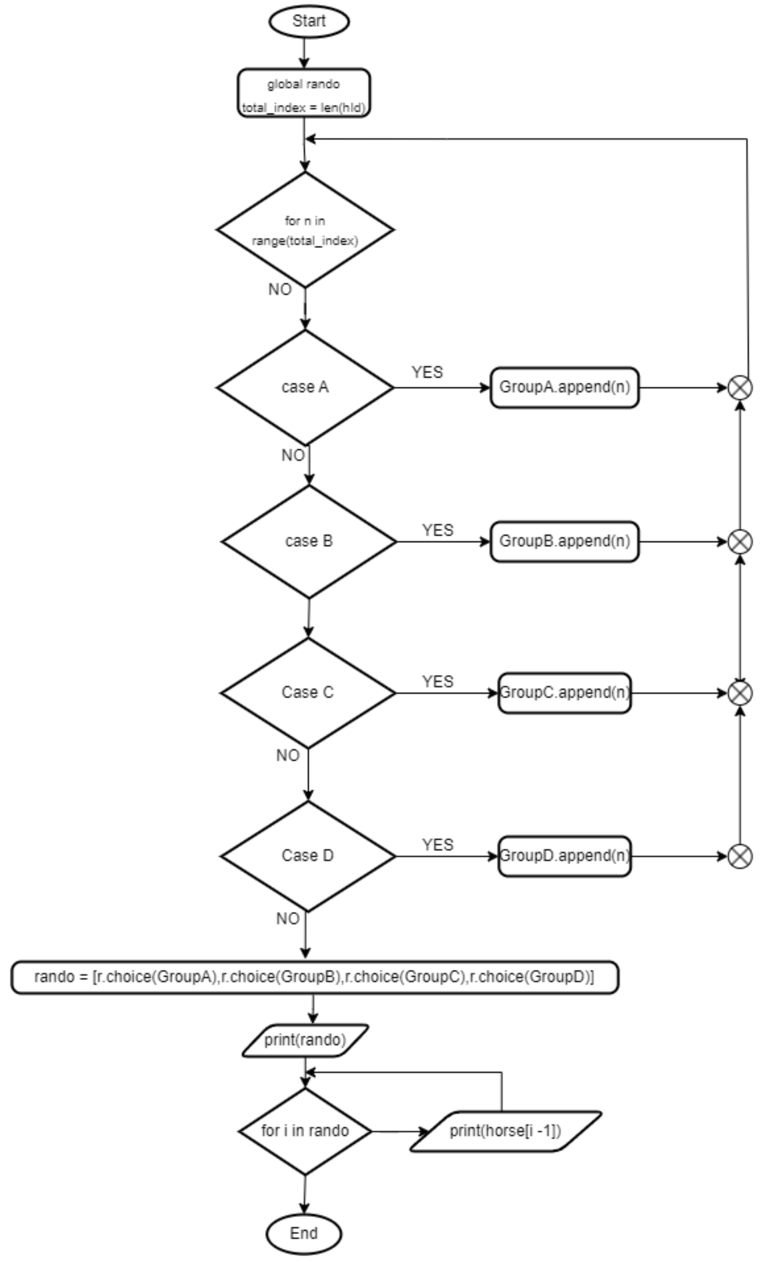
VHD() / View Horse Details


Start
header = " | horseId | horseName | horseJockey | horseAge | horseBreed | totalWin | totalRace | horseGroup | "
Print (' - ' * 136)
Print (header)
(' - ' *136)
READ the lines in the text file
for line in lines
Print each element in data with ' | ' operator
Finish

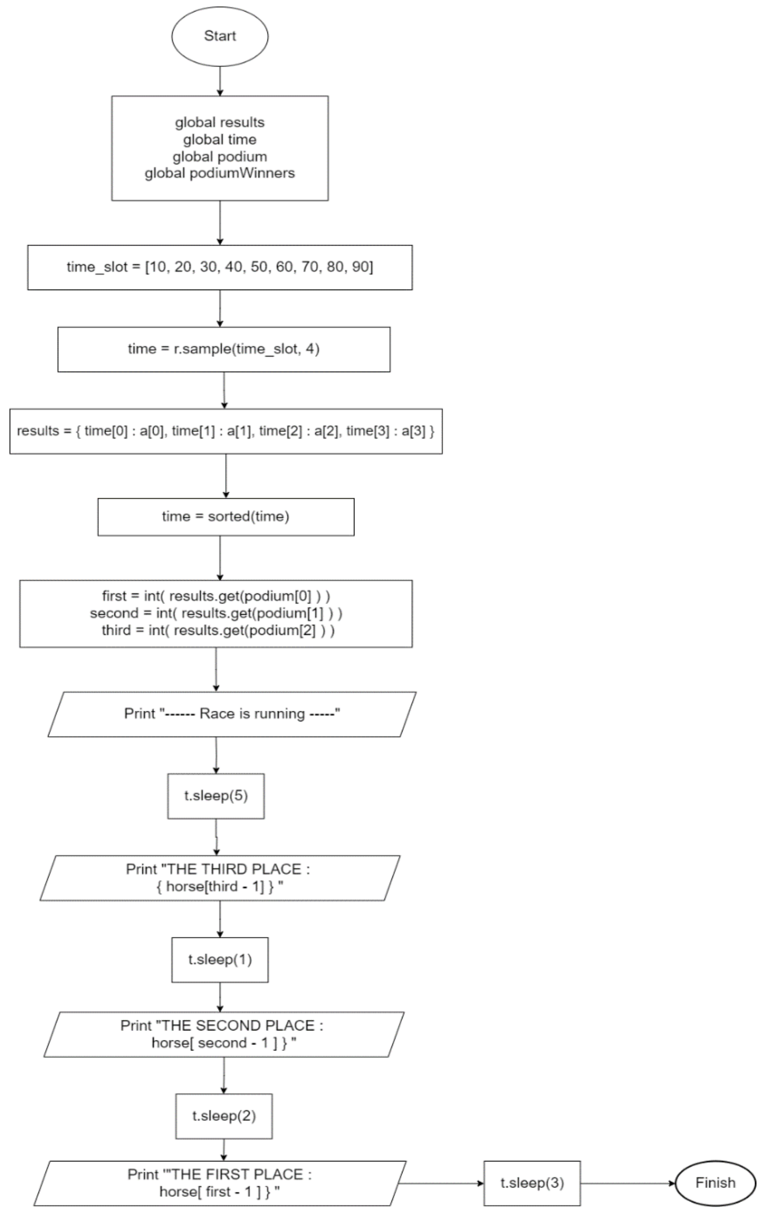
SHD() / Save Horse Details


Start
Print "YOU ARE GOING TO SAVE CHANGES YOU MADE"
Print " - To confirm Type : 'Y'/'y' | if you have second thoughts Type : 'N'/'n' "
Print "TYPE YOUR OPTION : "
Input y_n
match Input
Case 'Y' | 'y'
True
Open Horse.txt file
comma = ' , '
for record in horse :
Write the record into text file
False
Case'N' | 'n'
True
Print "Okay ! "
Finish

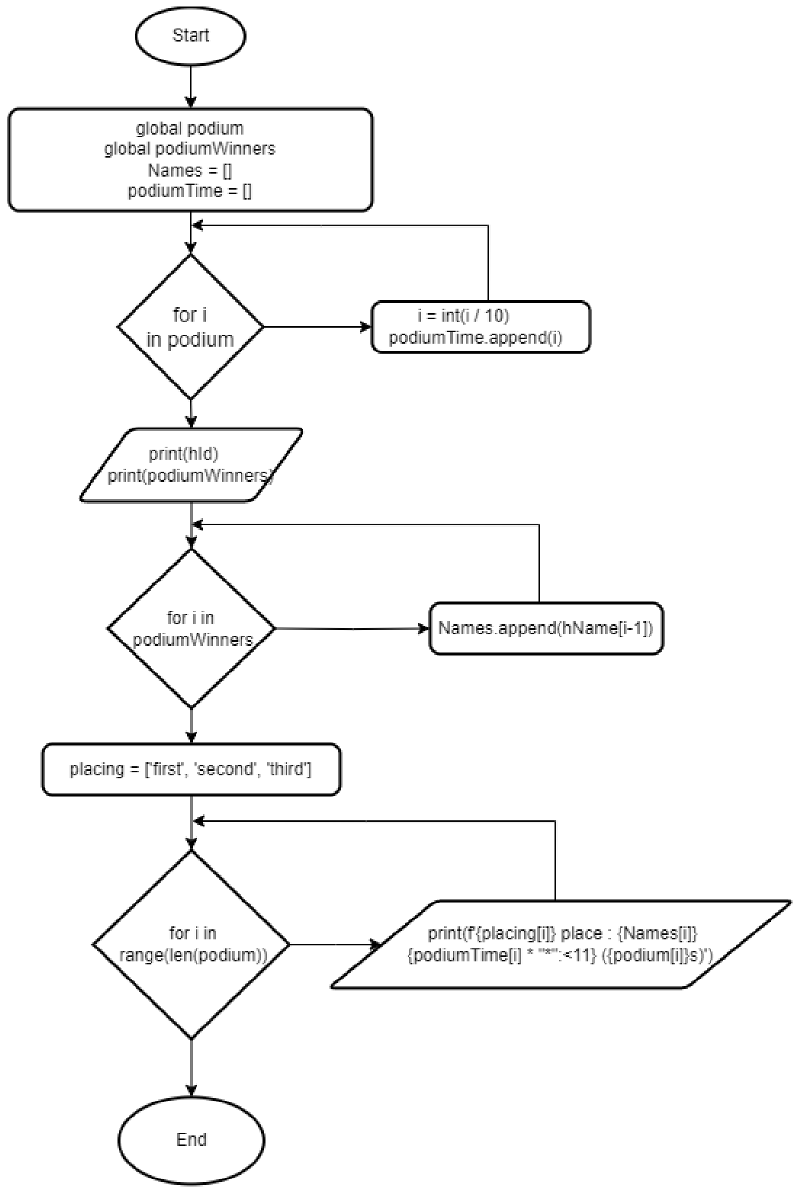
SDD() / Selecting four horses randomly



WHD() / Win Horse Details

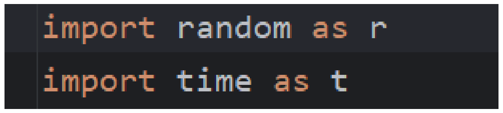

Start
global results
global time
global podium
global podiumWinners
time_slot = [10, 20, 30, 40, 50, 60, 70, 80, 90]
time = r.sample(time_slot, 4)
results = { time[0] : a[0], time[1] : a[1], time[2] : a[2], time[3] : a[3] }
time = sorted(time)
first = int(results.get(podium[0]))
second = int(results.get(podium[1]))
third = int(results.get(podium[2]))
Print "------ Race is running -----"
t.sleep(5)
Print "THE THIRD PLACE : { horse[third - 1] } "
t.sleep(1)
Print "THE SECOND PLACE : horse[second - 1] } "
t.sleep(2)
Print '"THE FIRST PLACE : horse[first - 1] } "
t.sleep(3)
Finish

VWH() / Display Winning horses



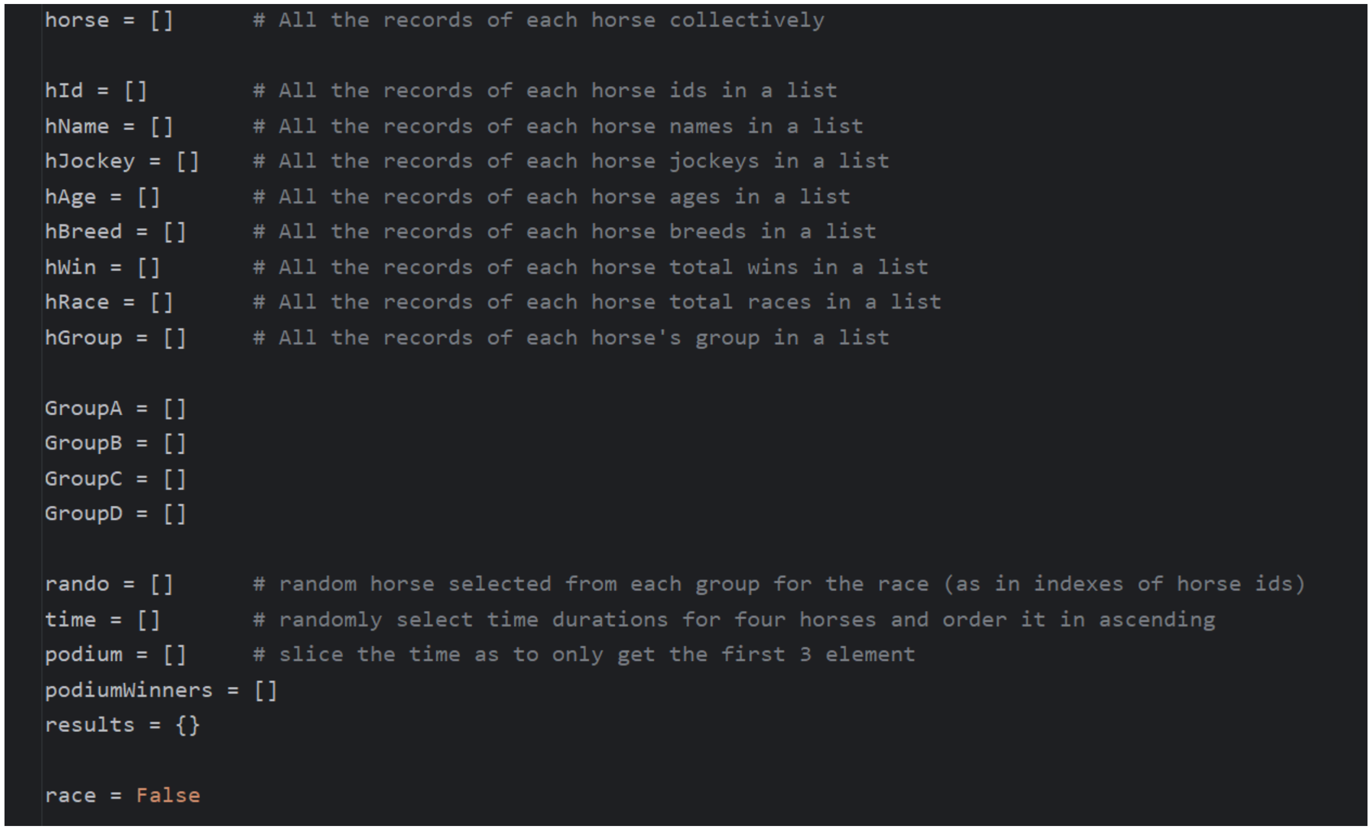
Code and Description

Libraries



```
import random as r
import time as t
```

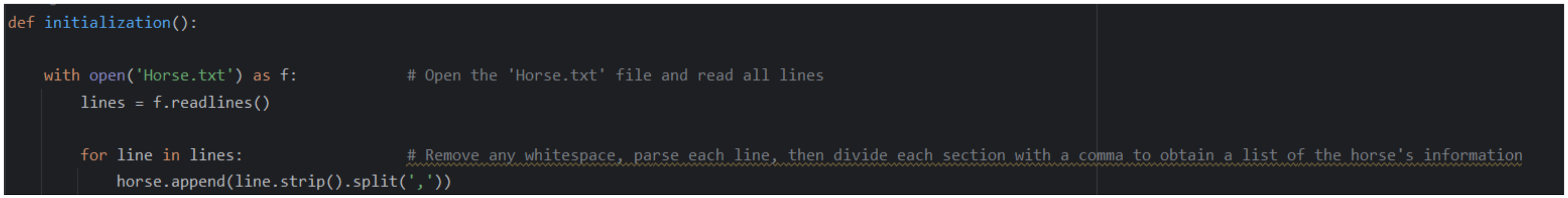
Variables and Arrays



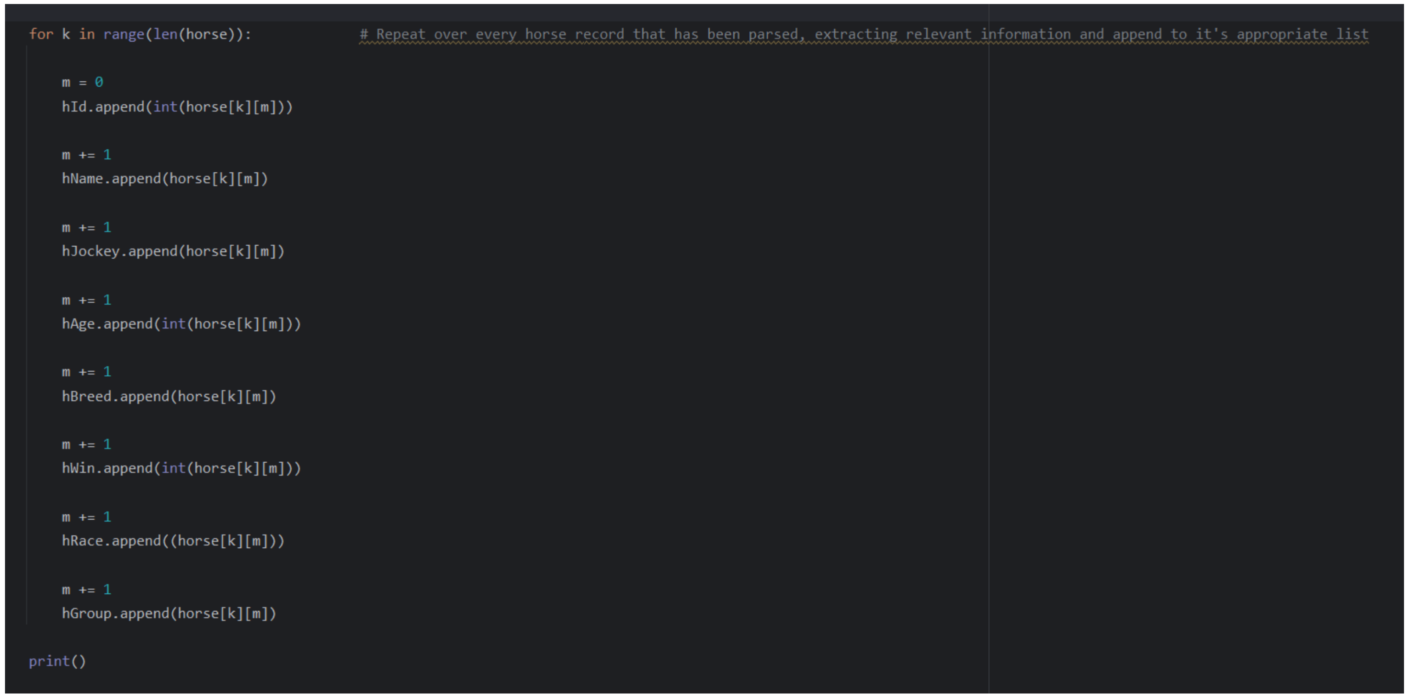
```
horse = []      # All the records of each horse collectively

hId = []        # All the records of each horse ids in a list
hName = []      # All the records of each horse names in a list
hJockey = []    # All the records of each horse jockeys in a list
hAge = []       # All the records of each horse ages in a list
hBreed = []     # All the records of each horse breeds in a list
hWin = []       # All the records of each horse total wins in a list
hRace = []      # All the records of each horse total races in a list
hGroup = []     # All the records of each horse's group in a list

GroupA = []
GroupB = []
GroupC = []
GroupD = []

rando = []      # random horse selected from each group for the race (as in indexes of horse ids)
time = []       # randomly select time durations for four horses and order it in ascending
podium = []     # slice the time as to only get the first 3 element
podiumWinners = []
results = {}

race = False
```

initialization()

This code initializes several lists (hId, hName, hJockey, etc.) by reading information from a file named 'Horse.txt'. It reads each line from the file, strips whitespace, and splits the values using commas to populate the lists with horse-related information such as ID, name, jockey, age, breed, wins, race, and group



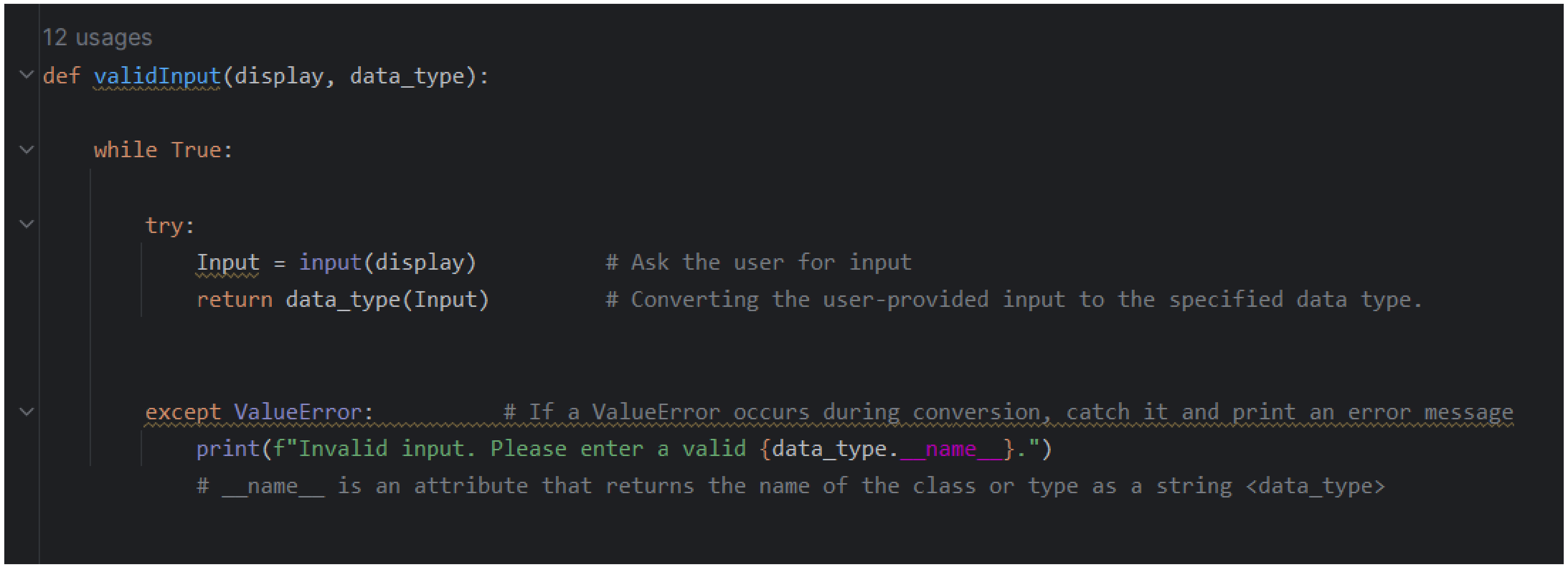
```
def initialization():

    with open('Horse.txt') as f:            # Open the 'Horse.txt' file and read all lines
        lines = f.readlines()

        for line in lines:                  # Remove any whitespace, parse each line, then divide each section with a comma to obtain a list of the horse's information
            horse.append(line.strip().split(','))
```



```
for k in range(len(horse)):            # Repeat over every horse record that has been parsed, extracting relevant information and append to it's appropriate list

    m = 0
    hId.append(int(horse[k][m]))

    m += 1
    hName.append(horse[k][m])

    m += 1
    hJockey.append(horse[k][m])

    m += 1
    hAge.append(int(horse[k][m]))

    m += 1
    hBreed.append(horse[k][m])

    m += 1
    hWin.append(int(horse[k][m]))

    m += 1
    hRace.append((horse[k][m]))

    m += 1
    hGroup.append(horse[k][m])

print()
```

validInput()

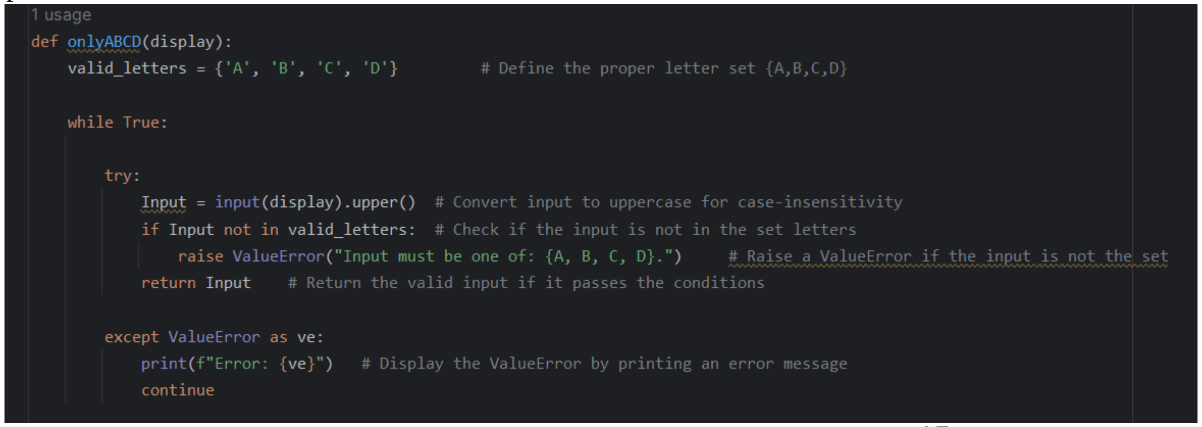
This code defines a function called validInput, which accepts as inputs a data type called data_type and a prompt message called display. In a loop, it asks the user for input multiple times, tries to convert it to the desired data type, and then outputs the transformed value. In the event that a ValueError arises during conversion, it prints an error notice, catches the exception, and asks the user for valid input once more.



```
12 usages
def validInput(display, data_type):

    while True:

        try:
            Input = input(display)          # Ask the user for input
            return data_type(Input)         # Converting the user-provided input to the specified data type.


        except ValueError:          # If a ValueError occurs during conversion, catch it and print an error message
            print(f"Invalid input. Please enter a valid {data_type.__name__}.")
            # __name__ is an attribute that returns the name of the class or type as a string <data_type>
```

onlyLetters()

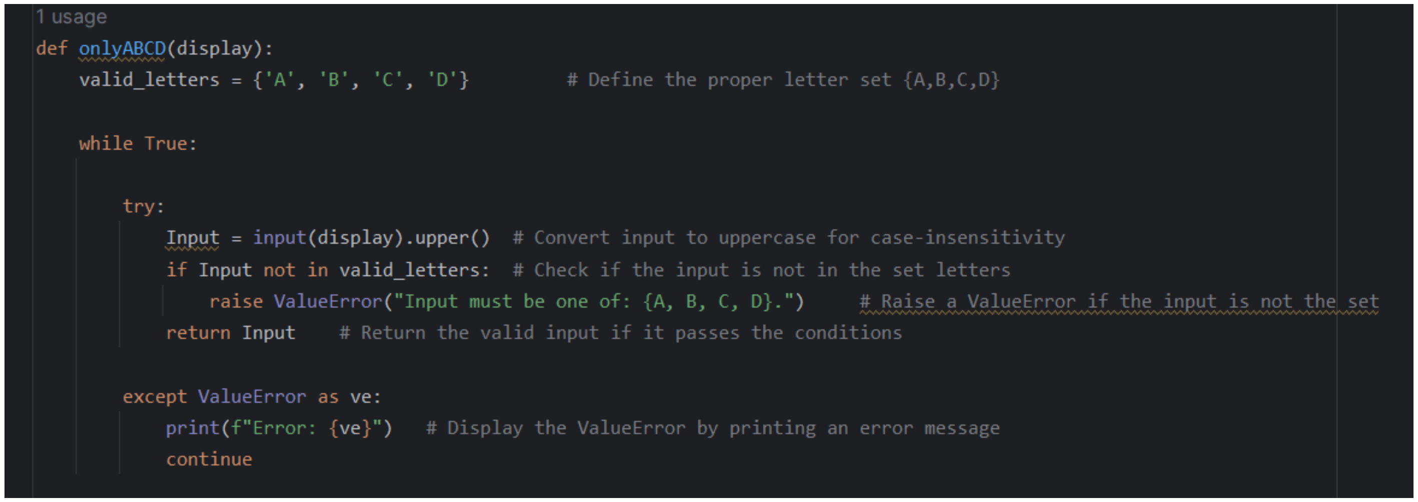
Within a loop, the function onlyLetters defined by this code asks the user for input. It determines whether the input is limited to alphabetical characters only; if not, it raises a ValueError along with an error message. The function returns the valid input when the user submits a valid input consisting solely of letters, ending the loop.



```
1 usage
def onlyABCD(display):
    valid_letters = {'A', 'B', 'C', 'D'}        # Define the proper letter set {A,B,C,D}

    while True:

        try:
            Input = input(display).upper()  # Convert input to uppercase for case-insensitivity
            if Input not in valid_letters:  # Check if the input is not in the set letters
                raise ValueError("Input must be one of: {A, B, C, D}.")     # Raise a ValueError if the input is not the set
            return Input    # Return the valid input if it passes the conditions

        except ValueError as ve:
            print(f"Error: {ve}")   # Display the ValueError by printing an error message
            continue
```

onlyABCD()

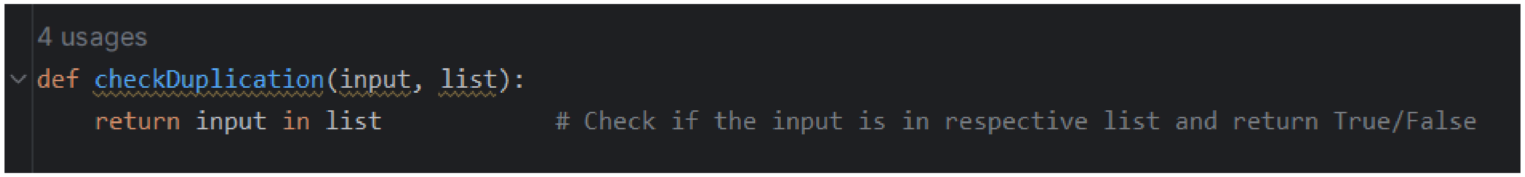
The function onlyABCD, which is defined in this code, asks the user for input repeatedly. It checks to see if the input is one of the case-insensitive characters "A," "B," "C," or "D." If it isn't, it raises a ValueError and displays an error message. The function returns the uppercase valid input after the loop keeps going till the user enters a valid input.



```
1 usage
def onlyABCD(display):
    valid_letters = {'A', 'B', 'C', 'D'}         # Define the proper letter set {A,B,C,D}

    while True:

        try:
            Input = input(display).upper()  # Convert input to uppercase for case-insensitivity
            if Input not in valid_letters:  # Check if the input is not in the set letters
                raise ValueError("Input must be one of: {A, B, C, D}.")     # Raise a ValueError if the input is not the set
            return Input    # Return the valid input if it passes the conditions

        except ValueError as ve:
            print(f"Error: {ve}")   # Display the ValueError by printing an error message
            continue
```

checkDuplication()

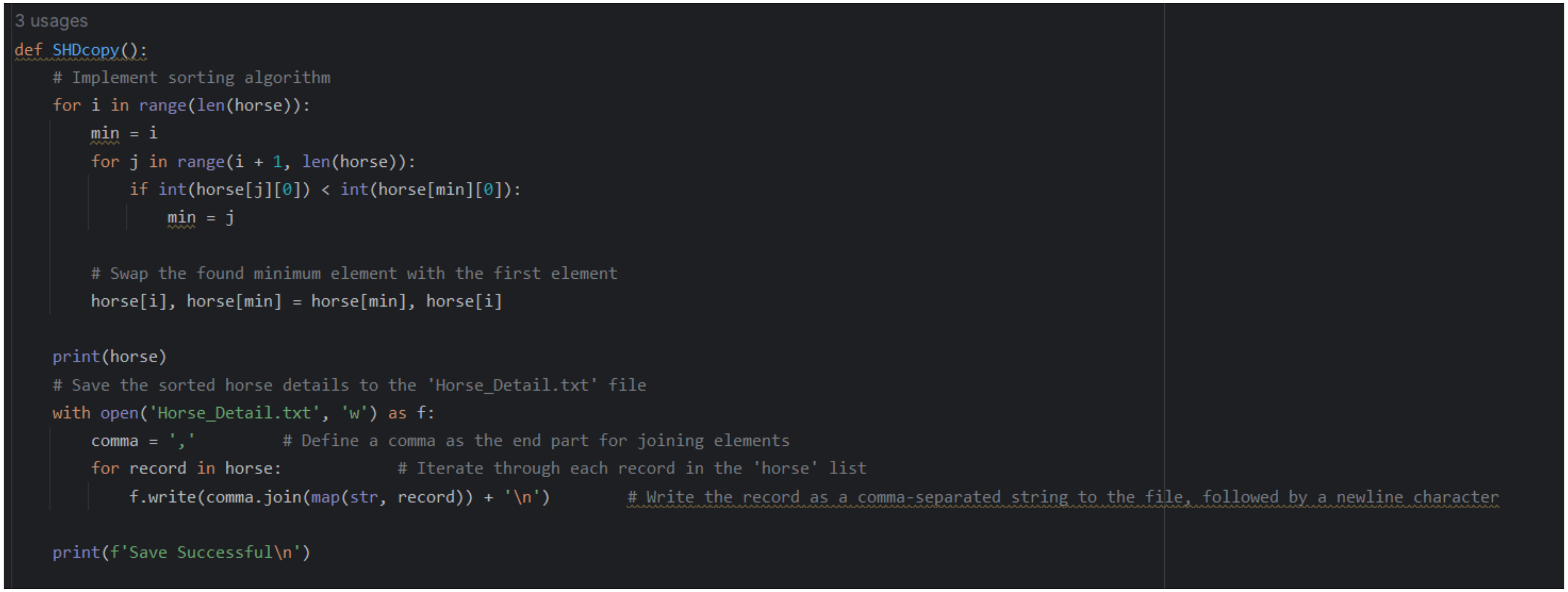
The checkDuplication function returns True if the given input is found in the specified list, indicating the presence of duplication; otherwise, it returns False



```
4 usages
def checkDuplication(input, list):
    return input in list            # Check if the input is in respective list and return True/False
```

SHDcopy()

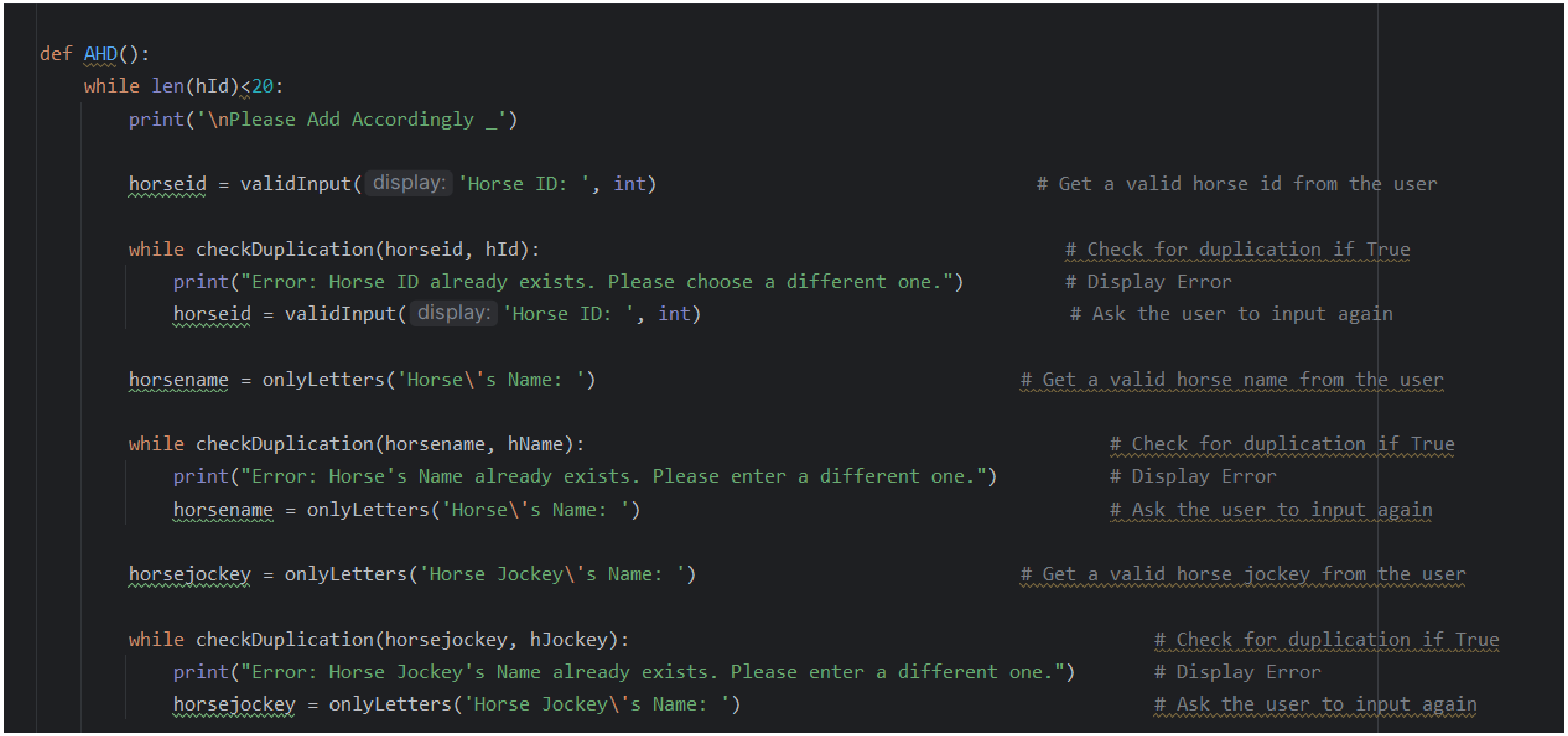
Based on the horse IDs, this code selects and sorts a list of horse records. After that, it publishes the sorted list of horse details in comma-separated format to a file called "Horse_Detail.txt," with each entry starting on a new line to signify a successful save.



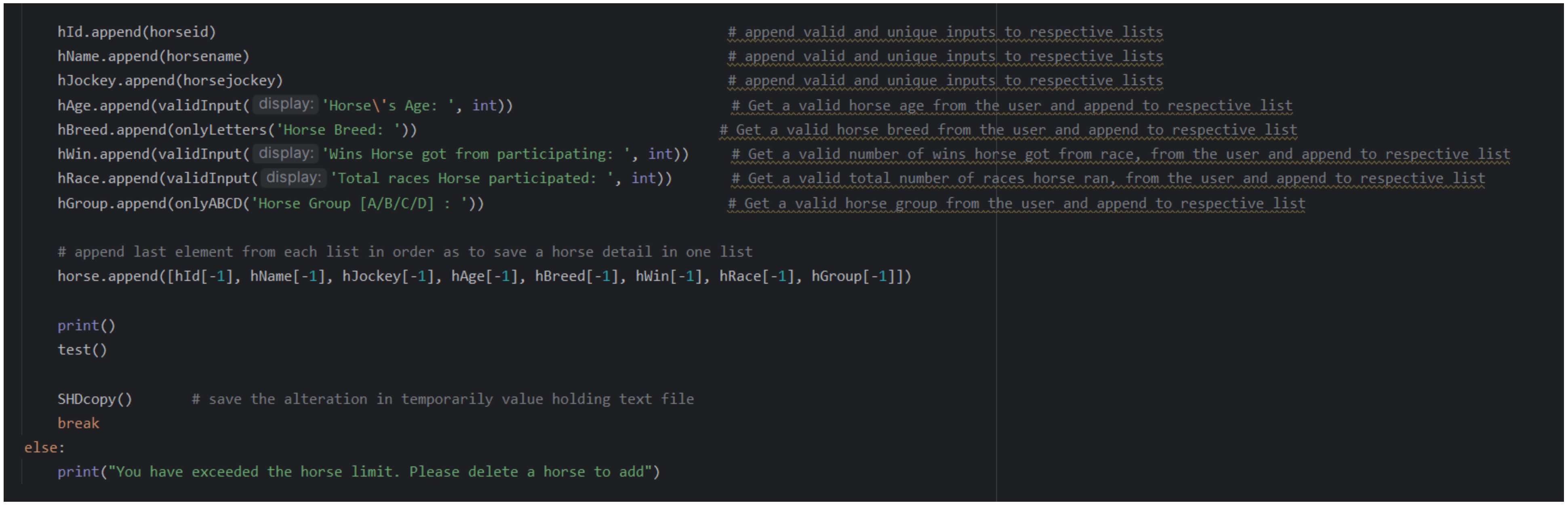
```
3 usages
def SHDcopy():
    # Implement sorting algorithm
    for i in range(len(horse)):
        min = i
        for j in range(i + 1, len(horse)):
            if int(horse[j][0]) < int(horse[min][0]):
                min = j

        # Swap the found minimum element with the first element
        horse[i], horse[min] = horse[min], horse[i]

    print(horse)
    # Save the sorted horse details to the 'Horse_Detail.txt' file
    with open('Horse_Detail.txt', 'w') as f:
        comma = ','         # Define a comma as the end part for joining elements
        for record in horse:            # Iterate through each record in the 'horse' list
            f.write(comma.join(map(str, record)) + '\n')        # Write the record as a comma-separated string to the file, followed by a newline character

    print(f'Save Successful\n')
```

AHD()

The provided code adds horse details to various lists (hId, hName, hJockey, hAge, hBreed, hWin, hRace, hGroup, horse). It uses loops to ensure that the entered values are valid and not duplicates, repeating the input process if necessary. If the maximum limit of 20 horses is reached, it informs the user that no more horses can be added until an existing one is deleted. Finally, it calls the SHDcopy function to save the updated details to a text file



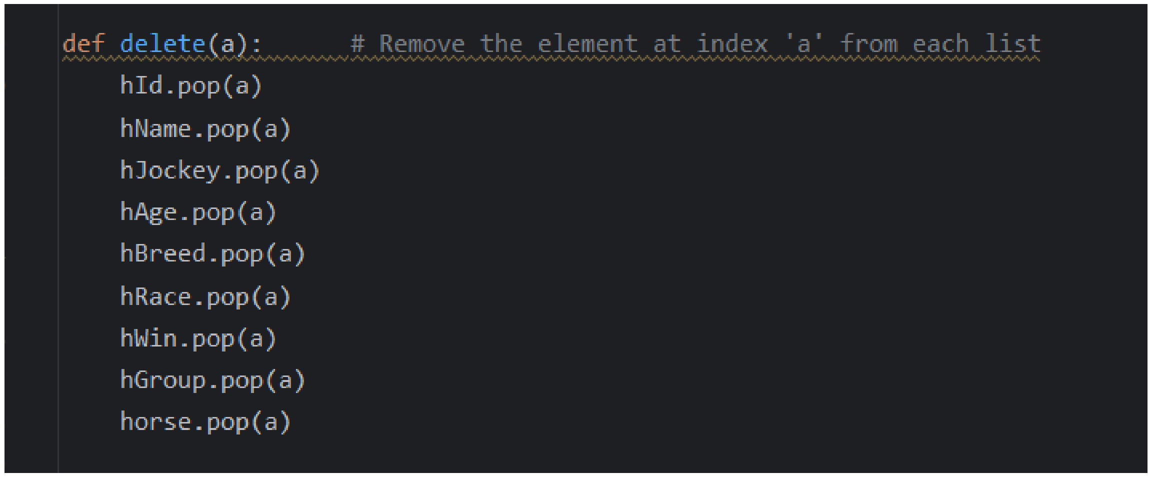
```
def AHD():
    while len(hId)<20:
        print('\nPlease Add Accordingly _')

        horseid = validInput( display: 'Horse ID: ', int)                              # Get a valid horse id from the user

        while checkDuplication(horseid, hId):                                           # Check for duplication if True
            print("Error: Horse ID already exists. Please choose a different one.")     # Display Error
            horseid = validInput( display: 'Horse ID: ', int)                           # Ask the user to input again

        horsename = onlyLetters('Horse\'s Name: ')                                     # Get a valid horse name from the user

        while checkDuplication(horsename, hName):                                       # Check for duplication if True
            print("Error: Horse's Name already exists. Please enter a different one.")  # Display Error
            horsename = onlyLetters('Horse\'s Name: ')                                  # Ask the user to input again

        horsejockey = onlyLetters('Horse Jockey\'s Name: ')                            # Get a valid horse jockey from the user

        while checkDuplication(horsejockey, hJockey):                                           # Check for duplication if True
            print("Error: Horse Jockey's Name already exists. Please enter a different one.")   # Display Error
            horsejockey = onlyLetters('Horse Jockey\'s Name: ')                                 # Ask the user to input again
```



```
        hId.append(horseid)                                                        # append valid and unique inputs to respective lists
        hName.append(horsename)                                                    # append valid and unique inputs to respective lists
        hJockey.append(horsejockey)                                                # append valid and unique inputs to respective lists
        hAge.append(validInput( display: 'Horse\'s Age: ', int))                    # Get a valid horse age from the user and append to respective list
        hBreed.append(onlyLetters('Horse Breed: '))                               # Get a valid horse breed from the user and append to respective list
        hWin.append(validInput( display: 'Wins Horse got from participating: ', int))  # Get a valid number of wins horse got from race, from the user and append to respective list
        hRace.append(validInput( display: 'Total races Horse participated: ', int))    # Get a valid total number of races horse ran, from the user and append to respective list
        hGroup.append(onlyABCD('Horse Group [A/B/C/D] : '))                        # Get a valid horse group from the user and append to respective list

        # append last element from each list in order as to save a horse detail in one list
        horse.append([hId[-1], hName[-1], hJockey[-1], hAge[-1], hBreed[-1], hWin[-1], hRace[-1], hGroup[-1]])

        print()
        test()

        SHDcopy()       # save the alteration in temporarily value holding text file
        break
    else:
        print("You have exceeded the horse limit. Please delete a horse to add")
```

detele()

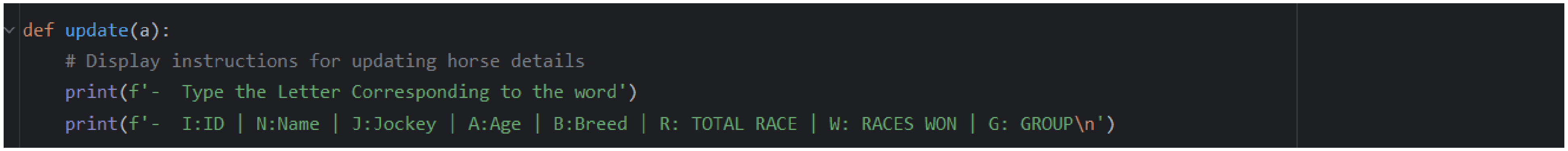
The elements at index 'a' from each corresponding list (hId, hName, hJockey, hAge, hBreed, hRace, hWin, hGroup, and horse) are eliminated by the delete function



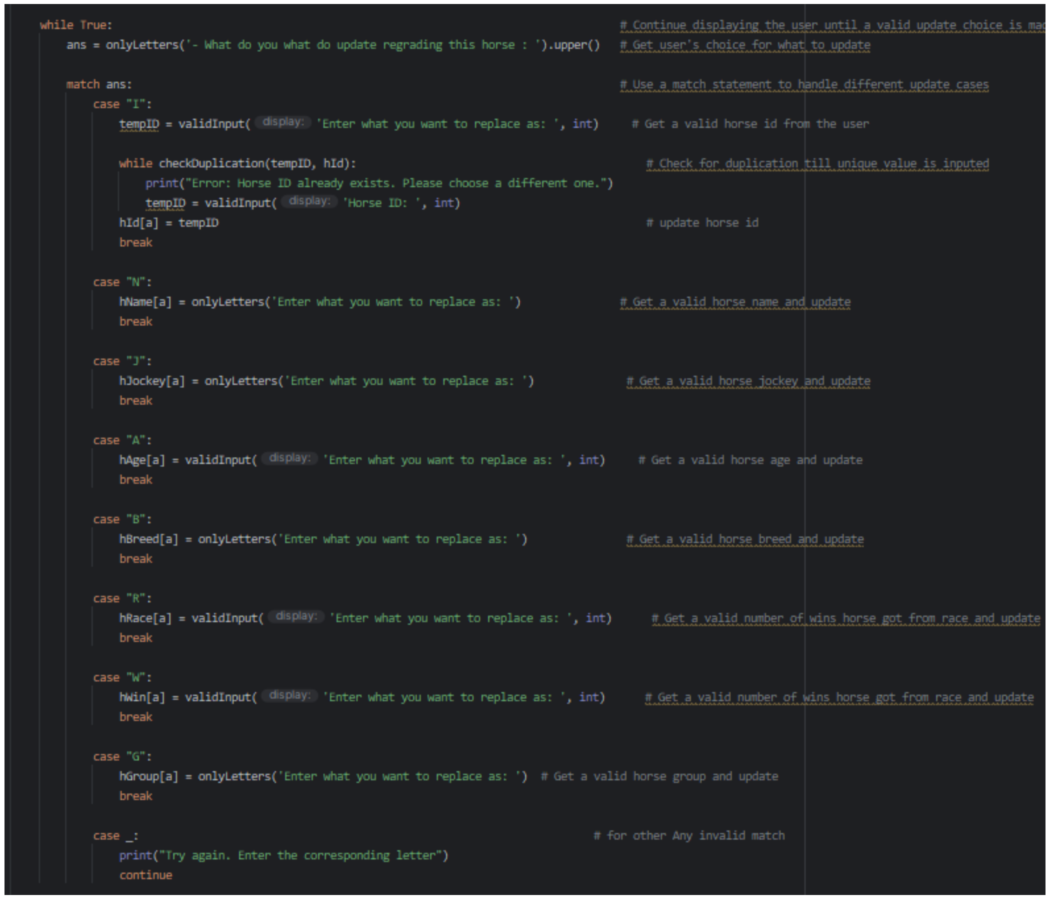
```
def delete(a):      # Remove the element at index 'a' from each list
    hId.pop(a)
    hName.pop(a)
    hJockey.pop(a)
    hAge.pop(a)
    hBreed.pop(a)
    hRace.pop(a)
    hWin.pop(a)
    hGroup.pop(a)
    horse.pop(a)
```

update()

The update function prompts the user to choose which aspect of a horse's details to update using a menu of options (ID, Name, Jockey, Age, Breed, Total Races, Races Won, Group). It then handles each update case based on the user's selection, ensuring the entered values are valid and, in the case of ID, unique by checking for duplication. The function uses a match statement to streamline the update process and continues prompting the user until a valid update choice is made.



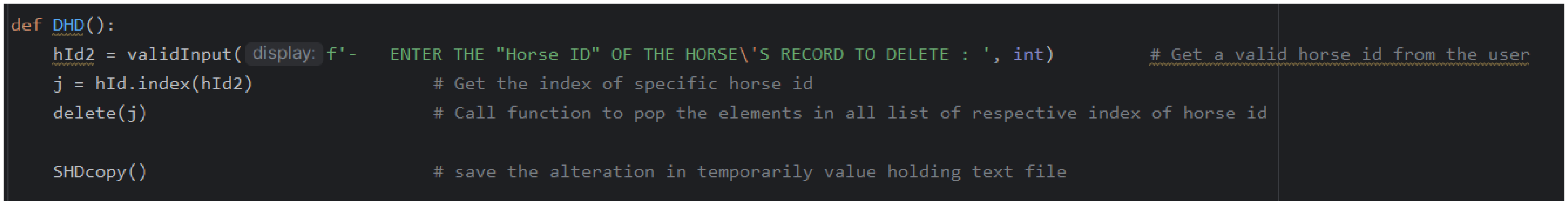
```python
def update(a):
    # Display instructions for updating horse details
    print(f'-  Type the Letter Corresponding to the word')
    print(f'-  I:ID | N:Name | J:Jockey | A:Age | B:Breed | R: TOTAL RACE | W: RACES WON | G: GROUP\n')
```



```python
    while True:                                                                    # Continue displaying the user until a valid update choice is mad
        ans = onlyLetters('- What do you what do update regrading this horse : ').upper()   # Get user's choice for what to update

        match ans:                                                                 # Use a match statement to handle different update cases
            case "I":
                tempID = validInput( display: 'Enter what you want to replace as: ', int)    # Get a valid horse id from the user

                while checkDuplication(tempID, hId):                                 # Check for duplication till unique value is inputed
                    print("Error: Horse ID already exists. Please choose a different one.")
                    tempID = validInput( display: 'Horse ID: ', int)
                hId[a] = tempID                                                      # update horse id
                break

            case "N":
                hName[a] = onlyLetters('Enter what you want to replace as: ')        # Get a valid horse name and update
                break

            case "J":
                hJockey[a] = onlyLetters('Enter what you want to replace as: ')      # Get a valid horse jockey and update
                break

            case "A":
                hAge[a] = validInput( display: 'Enter what you want to replace as: ', int)    # Get a valid horse age and update
                break

            case "B":
                hBreed[a] = onlyLetters('Enter what you want to replace as: ')       # Get a valid horse breed and update
                break

            case "R":
                hRace[a] = validInput( display: 'Enter what you want to replace as: ', int)     # Get a valid number of wins horse got from race and update
                break

            case "W":
                hWin[a] = validInput( display: 'Enter what you want to replace as: ', int)     # Get a valid number of wins horse got from race and update
                break

            case "G":
                hGroup[a] = onlyLetters('Enter what you want to replace as: ')  # Get a valid horse group and update
                break

            case _:                                                        # for other Any invalid match
                print("Try again. Enter the corresponding letter")
                continue
```

DHD()

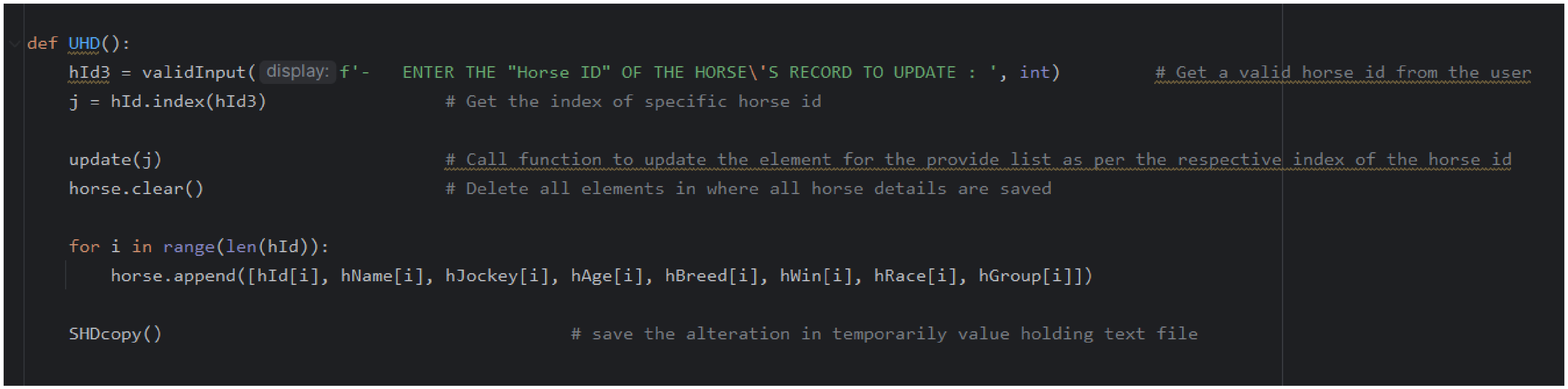
The DHD (Delete Horse Details) function prompts the user to enter the Horse ID of the record they want to delete. It then identifies the index of the specified Horse ID in the hId list, and subsequently, it calls the delete function to remove the corresponding elements at that index from all the respective lists, effectively deleting the entire record. Finally, the SHDcopy function is invoked to save the updated details to a text file named 'Horse_Detail.txt'.



```
def DHD():
    hId2 = validInput( display: f'-   ENTER THE "Horse ID" OF THE HORSE\'S RECORD TO DELETE : ', int)         # Get a valid horse id from the user
    j = hId.index(hId2)                    # Get the index of specific horse id
    delete(j)                              # Call function to pop the elements in all list of respective index of horse id

    SHDcopy()                              # save the alteration in temporarily value holding text file
```

UHD()

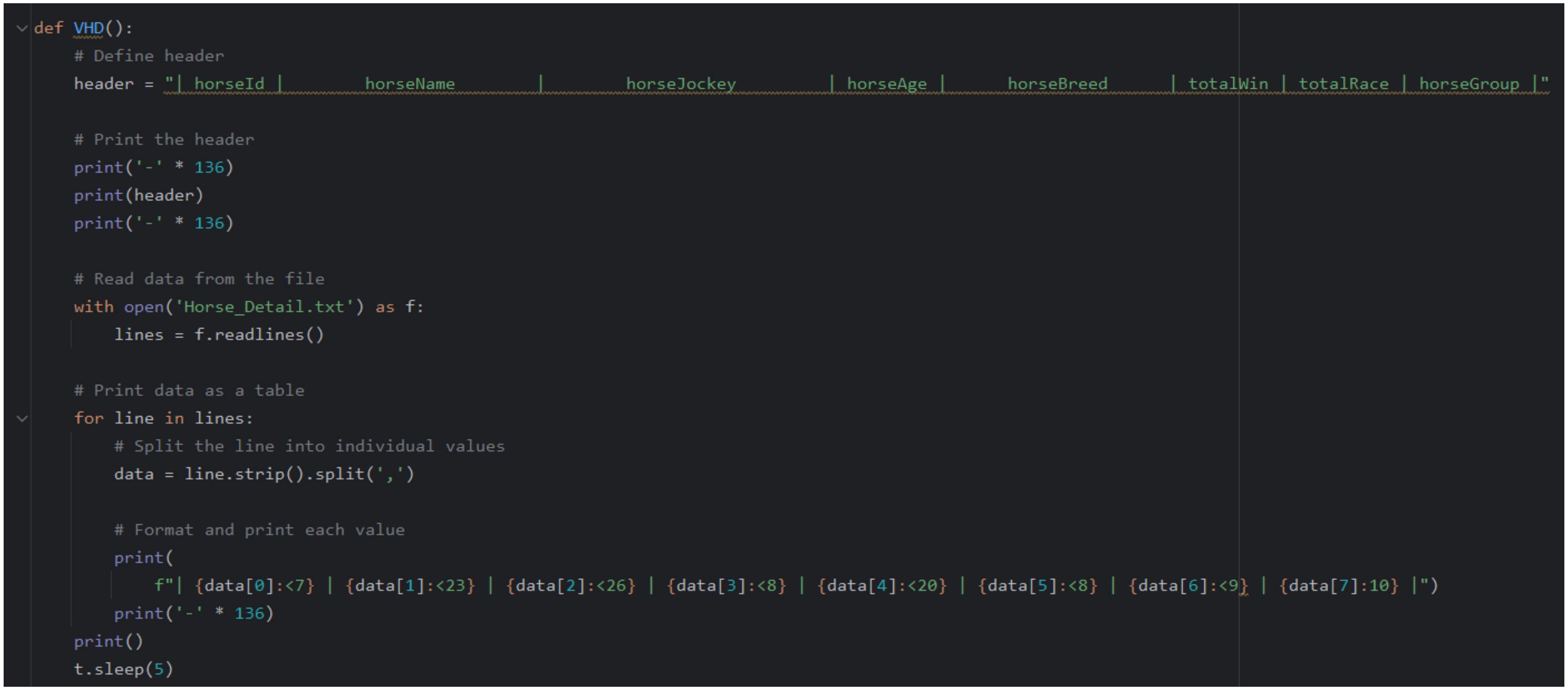
The UHD (Update Horse Details) function prompts the user to enter the Horse ID of the record they want to update. It then identifies the index of the specified Horse ID in the hId list and calls the update function to modify the details for the provided list at the respective index. After the update, it clears the horse list and reconstructs it by appending the updated details for all horses based on their indices in other lists. Finally, the SHDcopy function is invoked to save the updated details to a text file named 'Horse_Detail.txt'.



```
def UHD():
    hId3 = validInput( display: f'-   ENTER THE "Horse ID" OF THE HORSE\'S RECORD TO UPDATE : ', int)         # Get a valid horse id from the user
    j = hId.index(hId3)                  # Get the index of specific horse id

    update(j)                            # Call function to update the element for the provide list as per the respective index of the horse id
    horse.clear()                        # Delete all elements in where all horse details are saved

    for i in range(len(hId)):
        horse.append([hId[i], hName[i], hJockey[i], hAge[i], hBreed[i], hWin[i], hRace[i], hGroup[i]])

    SHDcopy()                                        # save the alteration in temporarily value holding text file
```

VHD()

The VHD (View Horse Details) function prints a formatted table of horse details by reading data from the 'Horse_Detail.txt' file. It begins by defining a header with specific column titles. After printing the header, the function reads each line from the file, splits it into individual values, and formats and prints each value in a tabular structure with appropriate column widths. The printed table includes details such as Horse ID, Name, Jockey, Age, Breed, Wins, Races, and Group. Finally, the function introduces a 5-second delay using time.sleep(5).

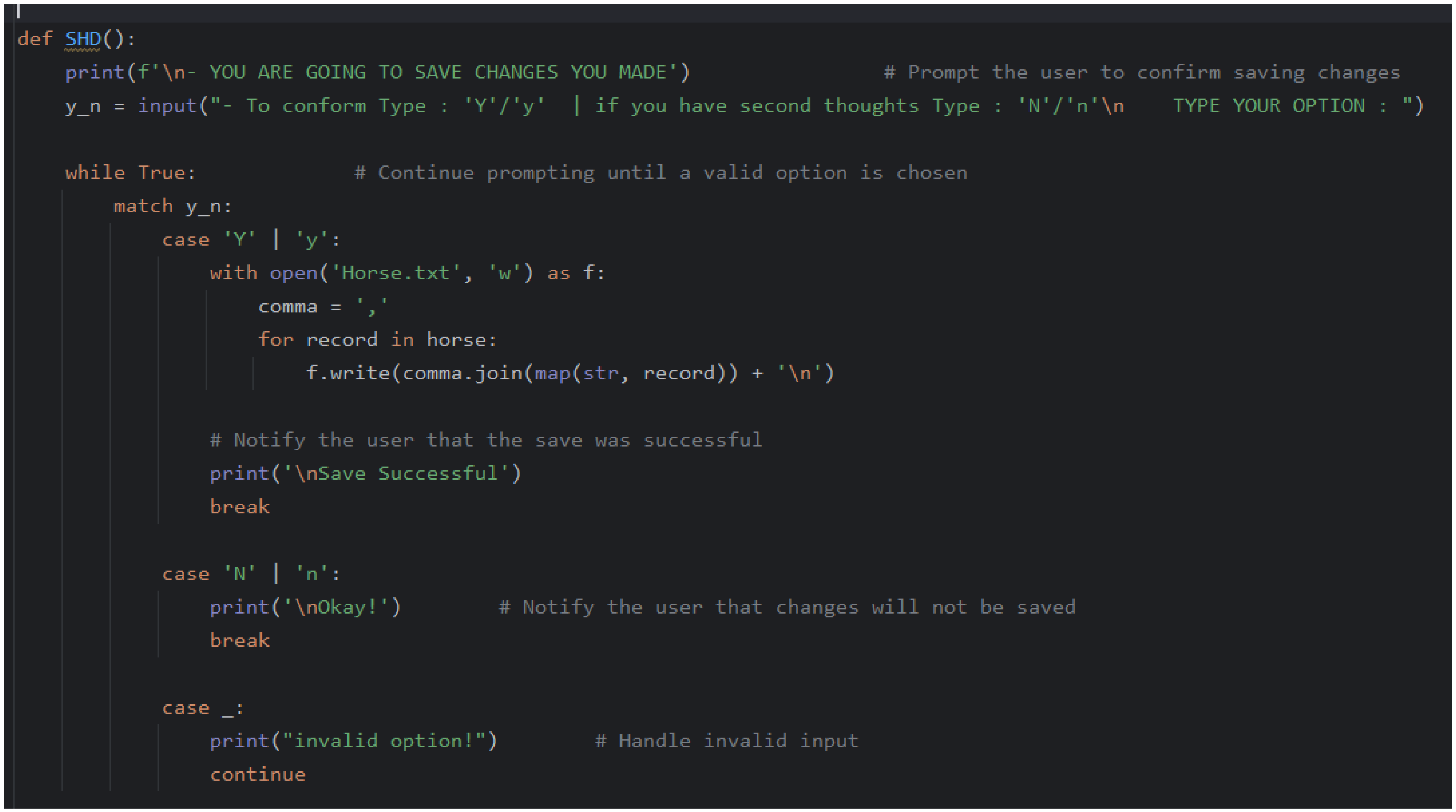


```python
def VHD():
    # Define header
    header = "| horseId |        horseName        |        horseJockey         | horseAge |      horseBreed      | totalWin | totalRace | horseGroup |"

    # Print the header
    print('-' * 136)
    print(header)
    print('-' * 136)

    # Read data from the file
    with open('Horse_Detail.txt') as f:
        lines = f.readlines()

    # Print data as a table
    for line in lines:
        # Split the line into individual values
        data = line.strip().split(',')

        # Format and print each value
        print(
            f"| {data[0]:<7} | {data[1]:<23} | {data[2]:<26} | {data[3]:<8} | {data[4]:<20} | {data[5]:<8} | {data[6]:<9} | {data[7]:10} |")
        print('-' * 136)
    print()
    t.sleep(5)
```

SHD()

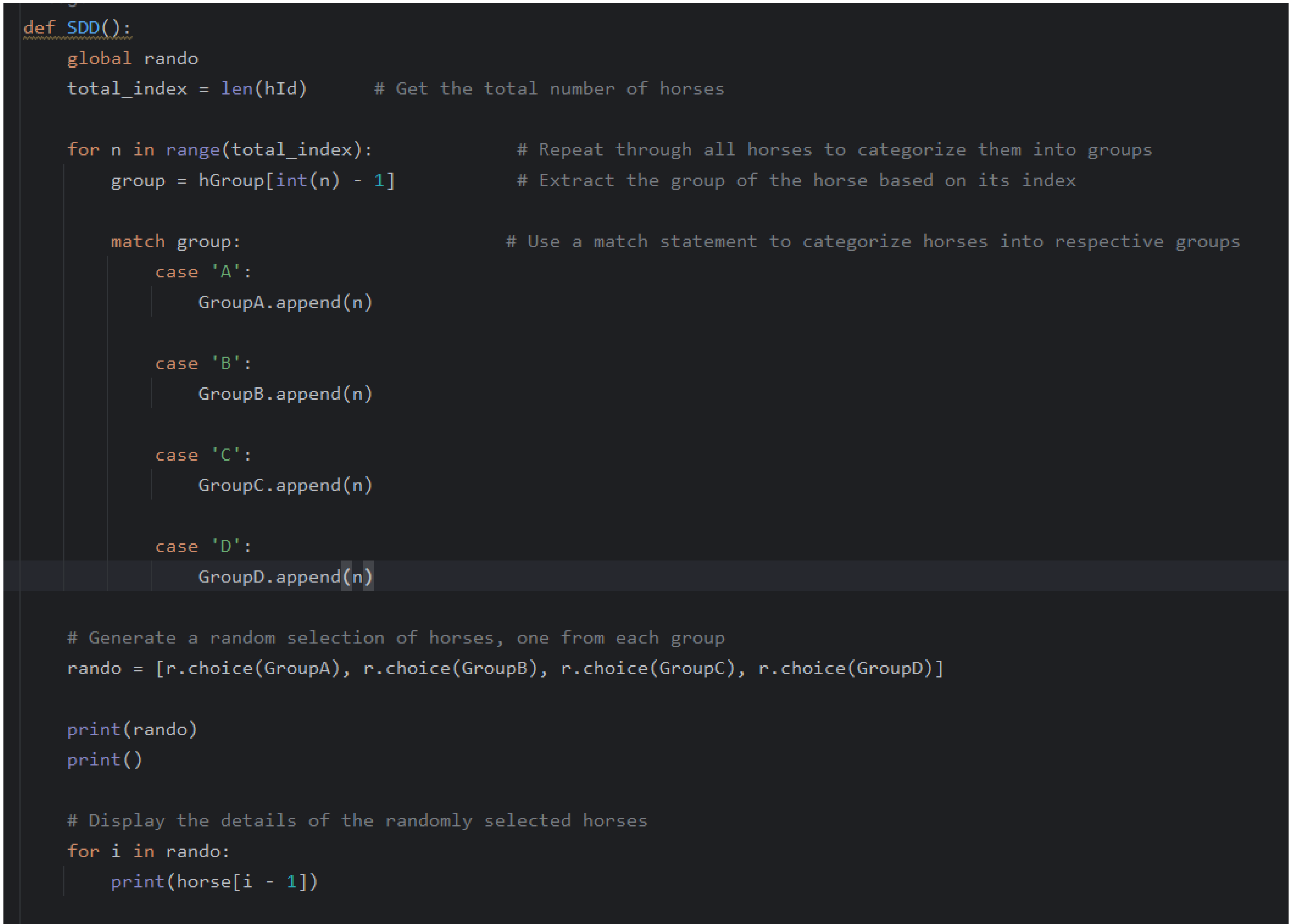
The SHD (Save Horse Details) function prompts the user to confirm whether they want to save the changes made to the horse details. It utilizes a match statement to handle user input, allowing 'Y' or 'y' to initiate the save process. If the user confirms the save, the function writes the updated horse details to the 'Horse.txt' file using a comma-separated format. It then notifies the user of the successful save, and if the user chooses not to save by entering 'N' or 'n', it simply breaks out of the loop

.



```python
def SHD():
    print(f'\n- YOU ARE GOING TO SAVE CHANGES YOU MADE')                # Prompt the user to confirm saving changes
    y_n = input("- To conform Type : 'Y'/'y'  | if you have second thoughts Type : 'N'/'n'\n    TYPE YOUR OPTION : ")

    while True:             # Continue prompting until a valid option is chosen
        match y_n:
            case 'Y' | 'y':
                with open('Horse.txt', 'w') as f:
                    comma = ','
                    for record in horse:
                        f.write(comma.join(map(str, record)) + '\n')

                # Notify the user that the save was successful
                print('\nSave Successful')
                break

            case 'N' | 'n':
                print('\nOkay!')        # Notify the user that changes will not be saved
                break

            case _:
                print("invalid option!")        # Handle invalid input
                continue
```

SDD()
The SDD function categorizes horses into groups (A, B, C, D) based on their assigned group values. It then randomly selects one horse from each group using the random.choice function and stores the indices in the rando list. Finally, it prints the indices of the randomly selected horses and displays their details by retrieving the information from the horse list using the stored indices. Note that there seems to be an off-by-one error in the index retrieval (horse[i - 1]), and the commented-out print statements can be used for debugging purposes.

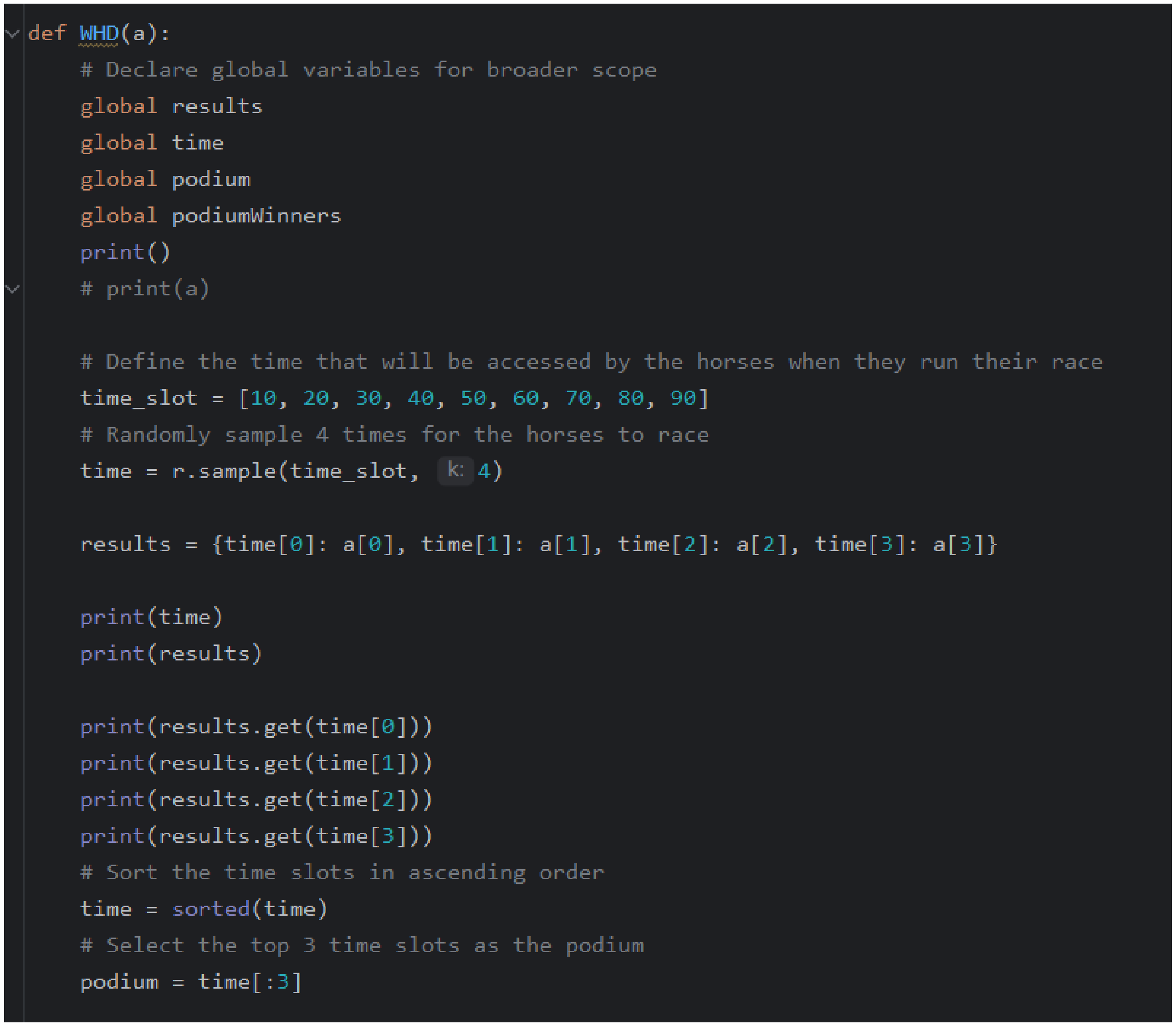


```python
def SDD():
    global rando
    total_index = len(hId)      # Get the total number of horses

    for n in range(total_index):             # Repeat through all horses to categorize them into groups
        group = hGroup[int(n) - 1]           # Extract the group of the horse based on its index

        match group:                        # Use a match statement to categorize horses into respective groups
            case 'A':
                GroupA.append(n)

            case 'B':
                GroupB.append(n)

            case 'C':
                GroupC.append(n)

            case 'D':
                GroupD.append(n)

    # Generate a random selection of horses, one from each group
    rando = [r.choice(GroupA), r.choice(GroupB), r.choice(GroupC), r.choice(GroupD)]

    print(rando)
    print()

    # Display the details of the randomly selected horses
    for i in rando:
        print(horse[i - 1])
```

WHD()

The WHD (Run Horse Race and Display Results) function simulates a horse race by randomly assigning time slots to four horses and storing the results in a dictionary. It then sorts the time slots in ascending order and selects the top three as the podium. The function retrieves the horse IDs corresponding to the first, second, and third places, stores them in the podiumWinners list, and prints the podium results with a delay between each announcement to simulate a race unfolding. The function seems to take a list of horse indices (a) as an argument and uses it to access horse details for displaying results

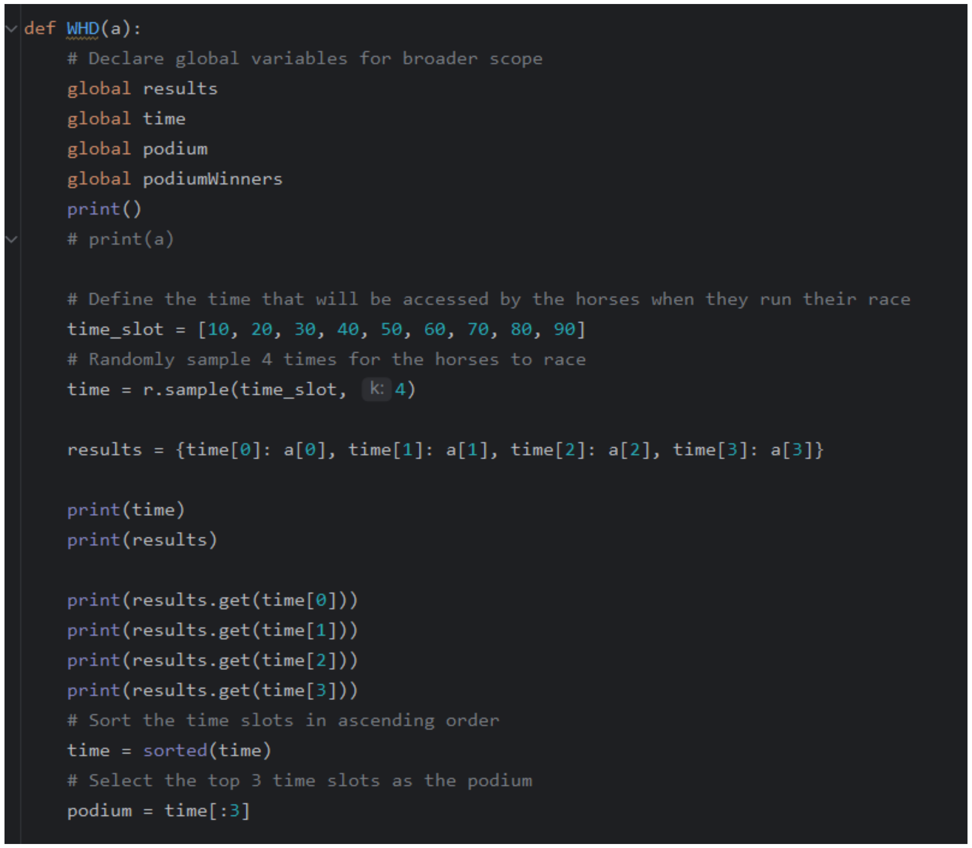
.



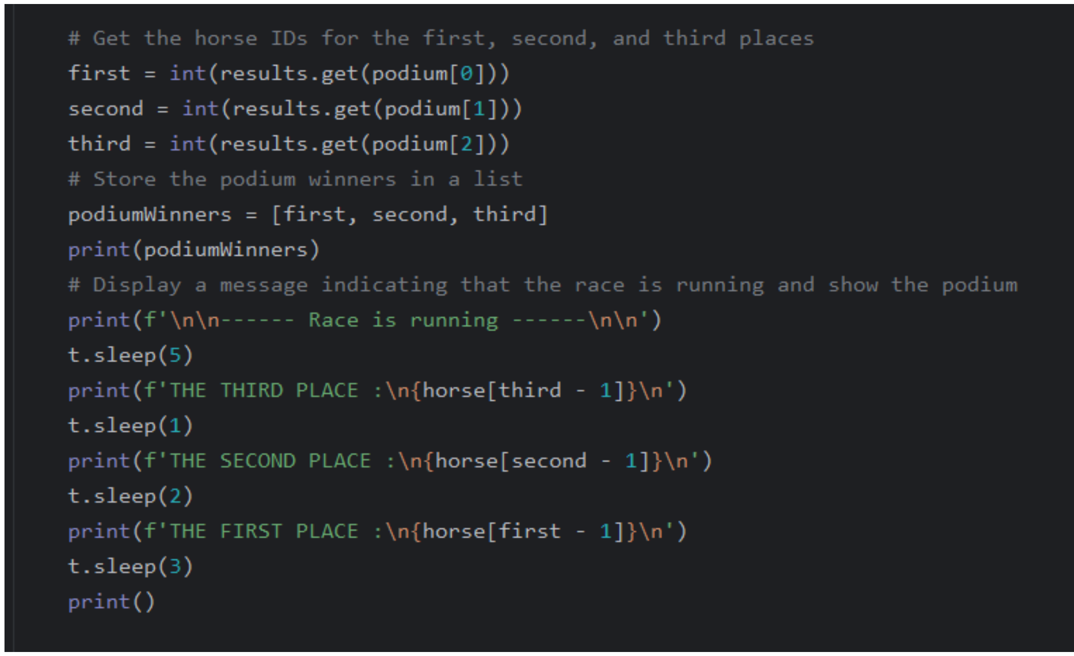
```python
def WHD(a):
    # Declare global variables for broader scope
    global results
    global time
    global podium
    global podiumWinners
    print()
    # print(a)

    # Define the time that will be accessed by the horses when they run their race
    time_slot = [10, 20, 30, 40, 50, 60, 70, 80, 90]
    # Randomly sample 4 times for the horses to race
    time = r.sample(time_slot,  k: 4)

    results = {time[0]: a[0], time[1]: a[1], time[2]: a[2], time[3]: a[3]}

    print(time)
    print(results)

    print(results.get(time[0]))
    print(results.get(time[1]))
    print(results.get(time[2]))
    print(results.get(time[3]))
    # Sort the time slots in ascending order
    time = sorted(time)
    # Select the top 3 time slots as the podium
    podium = time[:3]
```

VWH()

The VWH (Visualize Horse Race Results) function visualizes the final results of a horse race by converting the race times to star values and displaying the placing labels, horse names, and corresponding podium times. It uses the global variables podium, podiumWinners, hId, hName to retrieve and display relevant information. The function prints a formatted representation of the race results, including the placing labels ("first," "second," "third"), horse names, and visualized podium times.



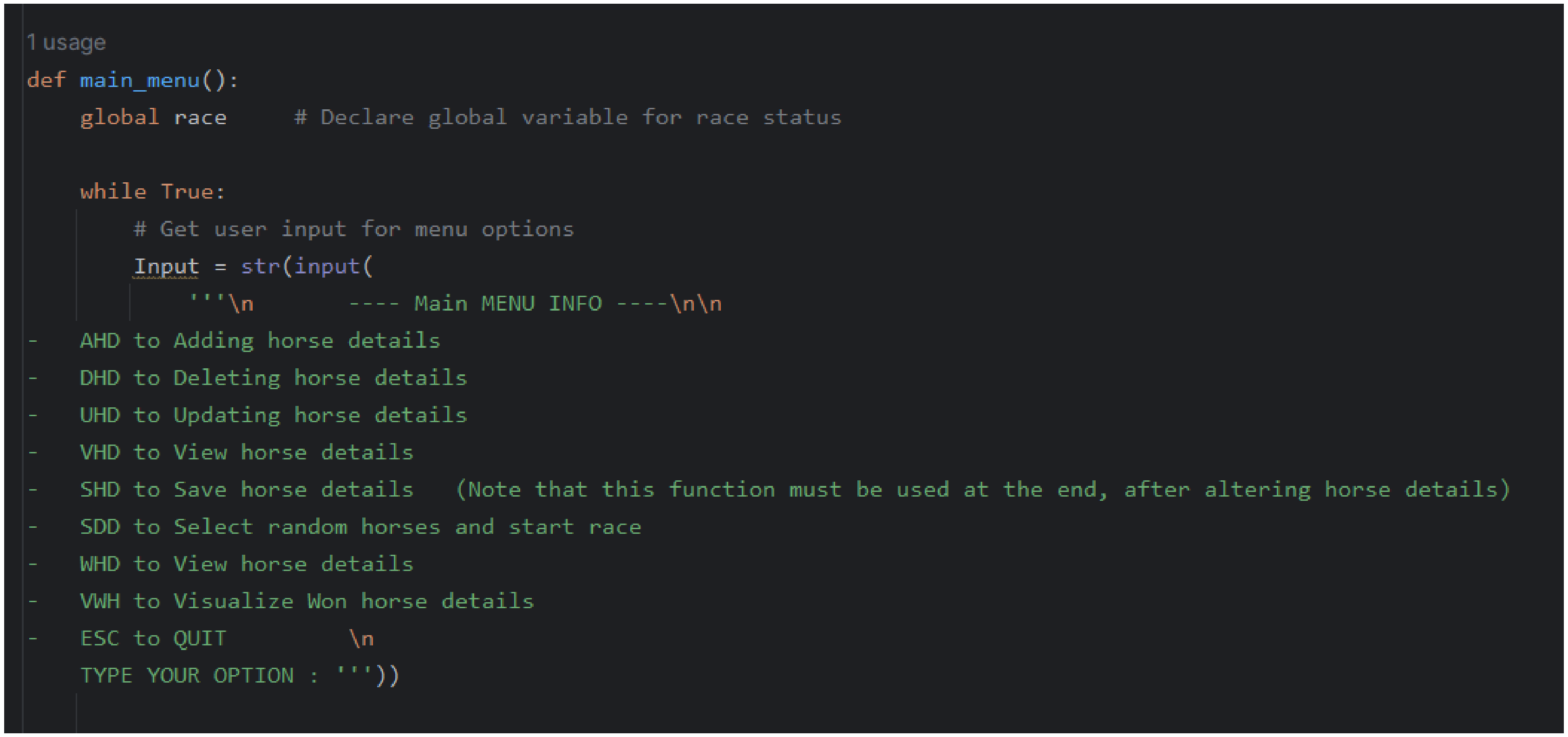
```python
def WHD(a):
    # Declare global variables for broader scope
    global results
    global time
    global podium
    global podiumWinners
    print()
    # print(a)

    # Define the time that will be accessed by the horses when they run their race
    time_slot = [10, 20, 30, 40, 50, 60, 70, 80, 90]
    # Randomly sample 4 times for the horses to race
    time = r.sample(time_slot, k: 4)

    results = {time[0]: a[0], time[1]: a[1], time[2]: a[2], time[3]: a[3]}

    print(time)
    print(results)

    print(results.get(time[0]))
    print(results.get(time[1]))
    print(results.get(time[2]))
    print(results.get(time[3]))
    # Sort the time slots in ascending order
    time = sorted(time)
    # Select the top 3 time slots as the podium
    podium = time[:3]
```



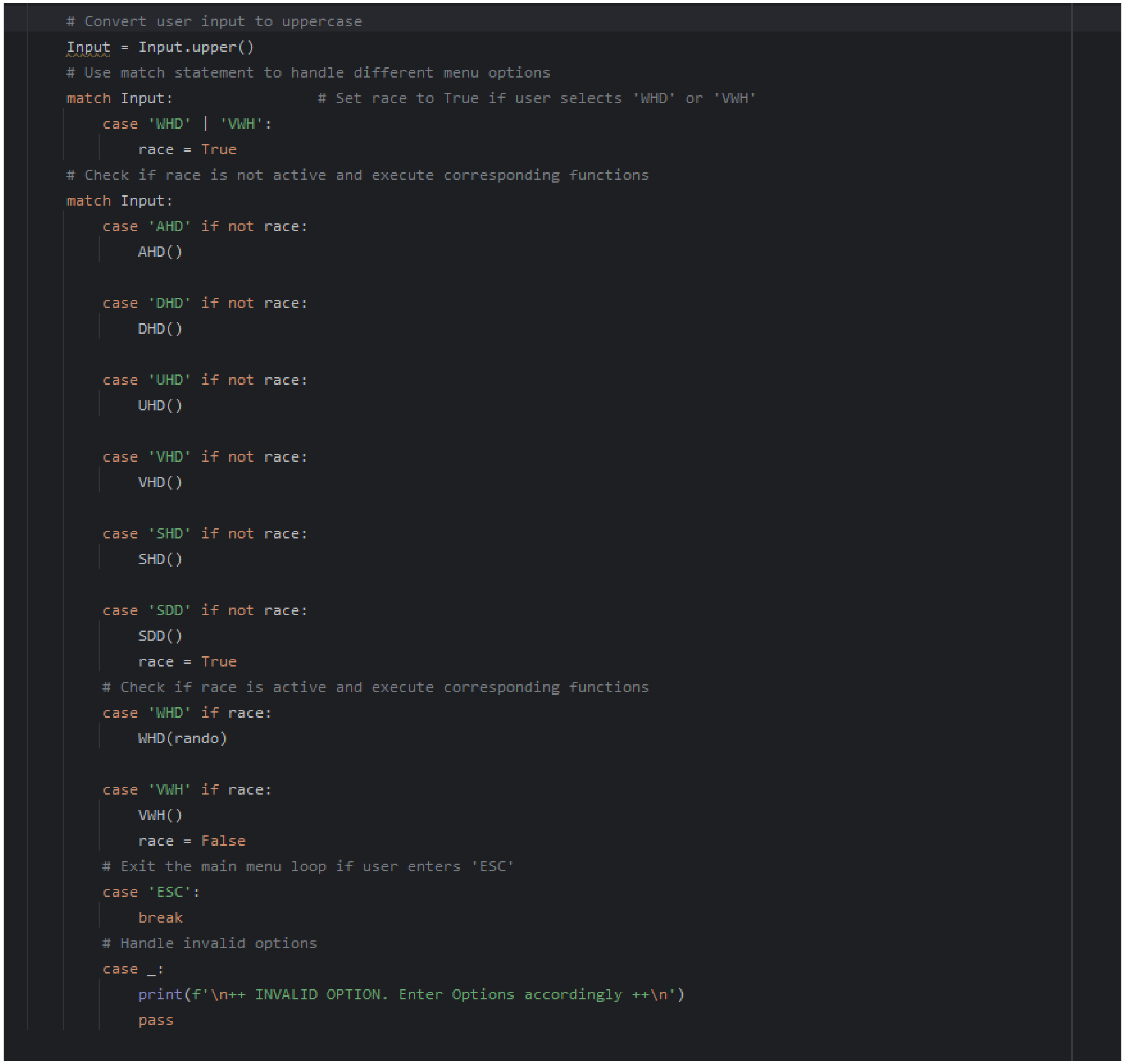
```python
    # Get the horse IDs for the first, second, and third places
    first = int(results.get(podium[0]))
    second = int(results.get(podium[1]))
    third = int(results.get(podium[2]))
    # Store the podium winners in a list
    podiumWinners = [first, second, third]
    print(podiumWinners)
    # Display a message indicating that the race is running and show the podium
    print(f'\n\n------ Race is running ------\n\n')
    t.sleep(5)
    print(f'THE THIRD PLACE :\n{horse[third - 1]}\n')
    t.sleep(1)
    print(f'THE SECOND PLACE :\n{horse[second - 1]}\n')
    t.sleep(2)
    print(f'THE FIRST PLACE :\n{horse[first - 1]}\n')
    t.sleep(3)
    print()
```

main_menu()

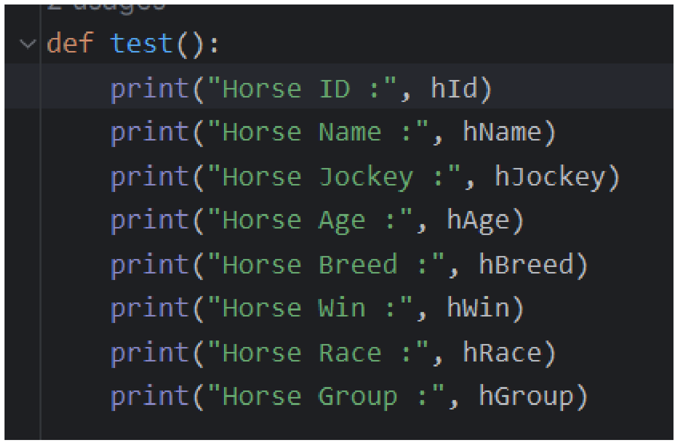
The main_menu function represents the main menu of the horse racing program. It repeatedly prompts the user to choose from various options, such as adding, deleting, updating, and viewing horse details, as well as initiating a horse race and visualizing the results. The function uses a match statement to handle different menu options and sets the race variable to True when the user selects options related to the horse race. It also checks the status of the race variable to determine whether certain menu options should be available or restricted. The loop continues until the user chooses the 'ESC' option to exit the program.



```
1 usage
def main_menu():
    global race     # Declare global variable for race status

    while True:
        # Get user input for menu options
        Input = str(input(
            '''\n       ---- Main MENU INFO ----\n\n
-   AHD to Adding horse details
-   DHD to Deleting horse details
-   UHD to Updating horse details
-   VHD to View horse details
-   SHD to Save horse details   (Note that this function must be used at the end, after altering horse details)
-   SDD to Select random horses and start race
-   WHD to View horse details
-   VWH to Visualize Won horse details
-   ESC to QUIT         \n
    TYPE YOUR OPTION : '''))
```

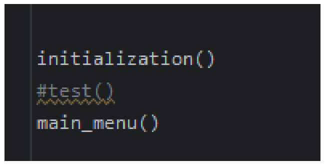


```
    # Convert user input to uppercase
    Input = Input.upper()
    # Use match statement to handle different menu options
    match Input:                # Set race to True if user selects 'WHD' or 'VWH'
        case 'WHD' | 'VWH':
            race = True
    # Check if race is not active and execute corresponding functions
    match Input:
        case 'AHD' if not race:
            AHD()

        case 'DHD' if not race:
            DHD()

        case 'UHD' if not race:
            UHD()

        case 'VHD' if not race:
            VHD()

        case 'SHD' if not race:
            SHD()

        case 'SDD' if not race:
            SDD()
            race = True
        # Check if race is active and execute corresponding functions
        case 'WHD' if race:
            WHD(rando)

        case 'VWH' if race:
            VWH()
            race = False
        # Exit the main menu loop if user enters 'ESC'
        case 'ESC':
            break
        # Handle invalid options
        case _:
            print(f'\n++ INVALID OPTION. Enter Options accordingly ++\n')
            pass
```



```
def test():
    print("Horse ID :", hId)
    print("Horse Name :", hName)
    print("Horse Jockey :", hJockey)
    print("Horse Age :", hAge)
    print("Horse Breed :", hBreed)
    print("Horse Win :", hWin)
    print("Horse Race :", hRace)
    print("Horse Group :", hGroup)
```

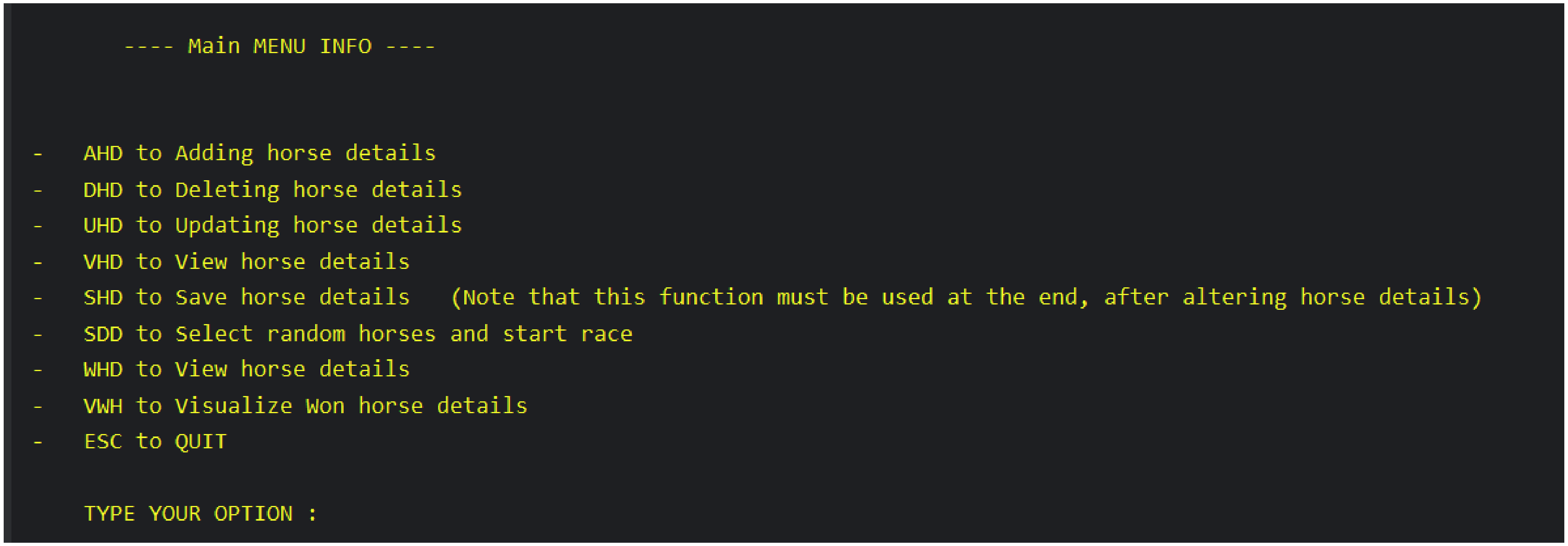
Run code



```
initialization()
#test()
main_menu()
```

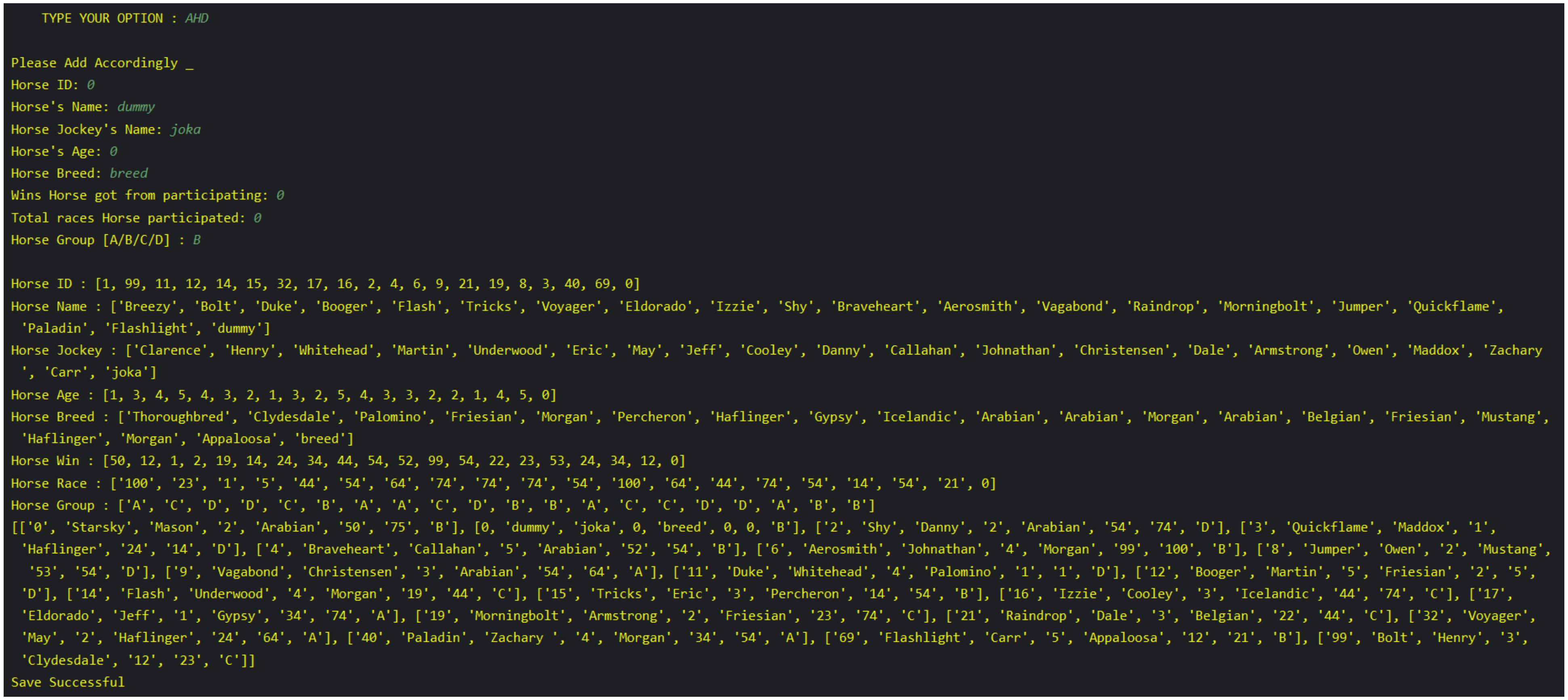
Output

Console Menu



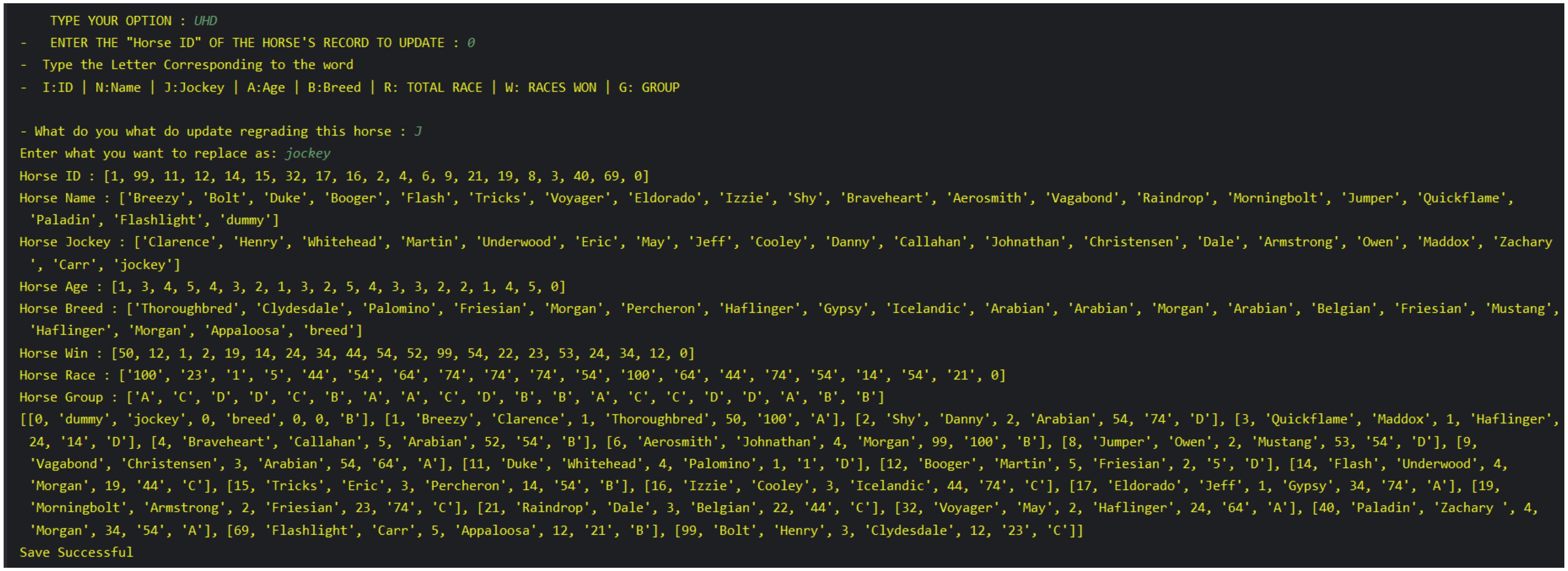
```
         ---- Main MENU INFO ----



 -   AHD to Adding horse details
 -   DHD to Deleting horse details
 -   UHD to Updating horse details
 -   VHD to View horse details
 -   SHD to Save horse details   (Note that this function must be used at the end, after altering horse details)
 -   SDD to Select random horses and start race
 -   WHD to View horse details
 -   VWH to Visualize Won horse details
 -   ESC to QUIT

     TYPE YOUR OPTION :
```

AHD



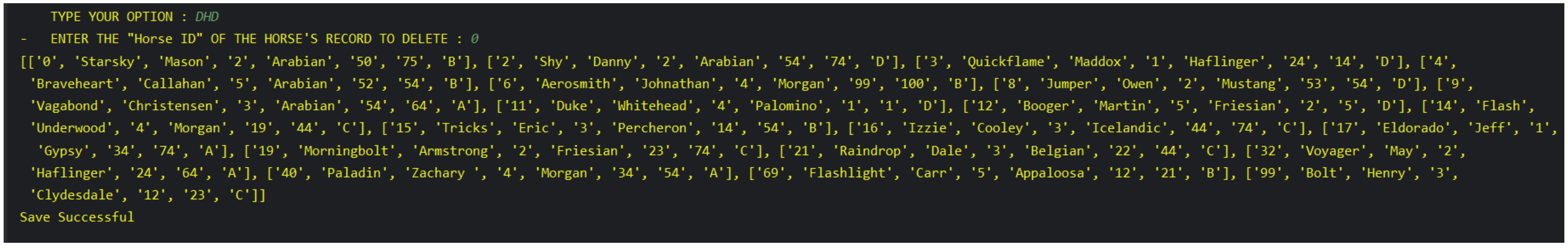
```
    TYPE YOUR OPTION : AHD

Please Add Accordingly _
Horse ID: 0
Horse's Name: dummy
Horse Jockey's Name: joka
Horse's Age: 0
Horse Breed: breed
Wins Horse got from participating: 0
Total races Horse participated: 0
Horse Group [A/B/C/D] : B

Horse ID : [1, 99, 11, 12, 14, 15, 32, 17, 16, 2, 4, 6, 9, 21, 19, 8, 3, 40, 69, 0]
Horse Name : ['Breezy', 'Bolt', 'Duke', 'Booger', 'Flash', 'Tricks', 'Voyager', 'Eldorado', 'Izzie', 'Shy', 'Braveheart', 'Aerosmith', 'Vagabond', 'Raindrop', 'Morningbolt', 'Jumper', 'Quickflame',
 'Paladin', 'Flashlight', 'dummy']
Horse Jockey : ['Clarence', 'Henry', 'Whitehead', 'Martin', 'Underwood', 'Eric', 'May', 'Jeff', 'Cooley', 'Danny', 'Callahan', 'Johnathan', 'Christensen', 'Dale', 'Armstrong', 'Owen', 'Maddox', 'Zachary
 ', 'Carr', 'joka']
Horse Age : [1, 3, 4, 5, 4, 3, 2, 1, 3, 2, 5, 4, 3, 3, 2, 2, 1, 4, 5, 0]
Horse Breed : ['Thoroughbred', 'Clydesdale', 'Palomino', 'Friesian', 'Morgan', 'Percheron', 'Haflinger', 'Gypsy', 'Icelandic', 'Arabian', 'Arabian', 'Morgan', 'Arabian', 'Belgian', 'Friesian', 'Mustang',
 'Haflinger', 'Morgan', 'Appaloosa', 'breed']
Horse Win : [50, 12, 1, 2, 19, 14, 24, 34, 44, 54, 52, 99, 54, 22, 23, 53, 24, 34, 12, 0]
Horse Race : ['100', '23', '1', '5', '44', '54', '64', '74', '74', '74', '54', '100', '64', '44', '74', '54', '14', '54', '21', 0]
Horse Group : ['A', 'C', 'D', 'D', 'C', 'B', 'A', 'A', 'C', 'D', 'B', 'B', 'A', 'C', 'C', 'D', 'D', 'A', 'B', 'B']
[['0', 'Starsky', 'Mason', '2', 'Arabian', '50', '75', 'B'], [0, 'dummy', 'joka', 0, 'breed', 0, 0, 'B'], ['2', 'Shy', 'Danny', '2', 'Arabian', '54', '74', 'D'], ['3', 'Quickflame', 'Maddox', '1',
 'Haflinger', '24', '14', 'D'], ['4', 'Braveheart', 'Callahan', '5', 'Arabian', '52', '54', 'B'], ['6', 'Aerosmith', 'Johnathan', '4', 'Morgan', '99', '100', 'B'], ['8', 'Jumper', 'Owen', '2', 'Mustang',
  '53', '54', 'D'], ['9', 'Vagabond', 'Christensen', '3', 'Arabian', '54', '64', 'A'], ['11', 'Duke', 'Whitehead', '4', 'Palomino', '1', '1', 'D'], ['12', 'Booger', 'Martin', '5', 'Friesian', '2', '5',
 'D'], ['14', 'Flash', 'Underwood', '4', 'Morgan', '19', '44', 'C'], ['15', 'Tricks', 'Eric', '3', 'Percheron', '14', '54', 'B'], ['16', 'Izzie', 'Cooley', '3', 'Icelandic', '44', '74', 'C'], ['17',
 'Eldorado', 'Jeff', '1', 'Gypsy', '34', '74', 'A'], ['19', 'Morningbolt', 'Armstrong', '2', 'Friesian', '23', '74', 'C'], ['21', 'Raindrop', 'Dale', '3', 'Belgian', '22', '44', 'C'], ['32', 'Voyager',
 'May', '2', 'Haflinger', '24', '64', 'A'], ['40', 'Paladin', 'Zachary ', '4', 'Morgan', '34', '54', 'A'], ['69', 'Flashlight', 'Carr', '5', 'Appaloosa', '12', '21', 'B'], ['99', 'Bolt', 'Henry', '3',
 'Clydesdale', '12', '23', 'C']]
Save Successful
```

UHD



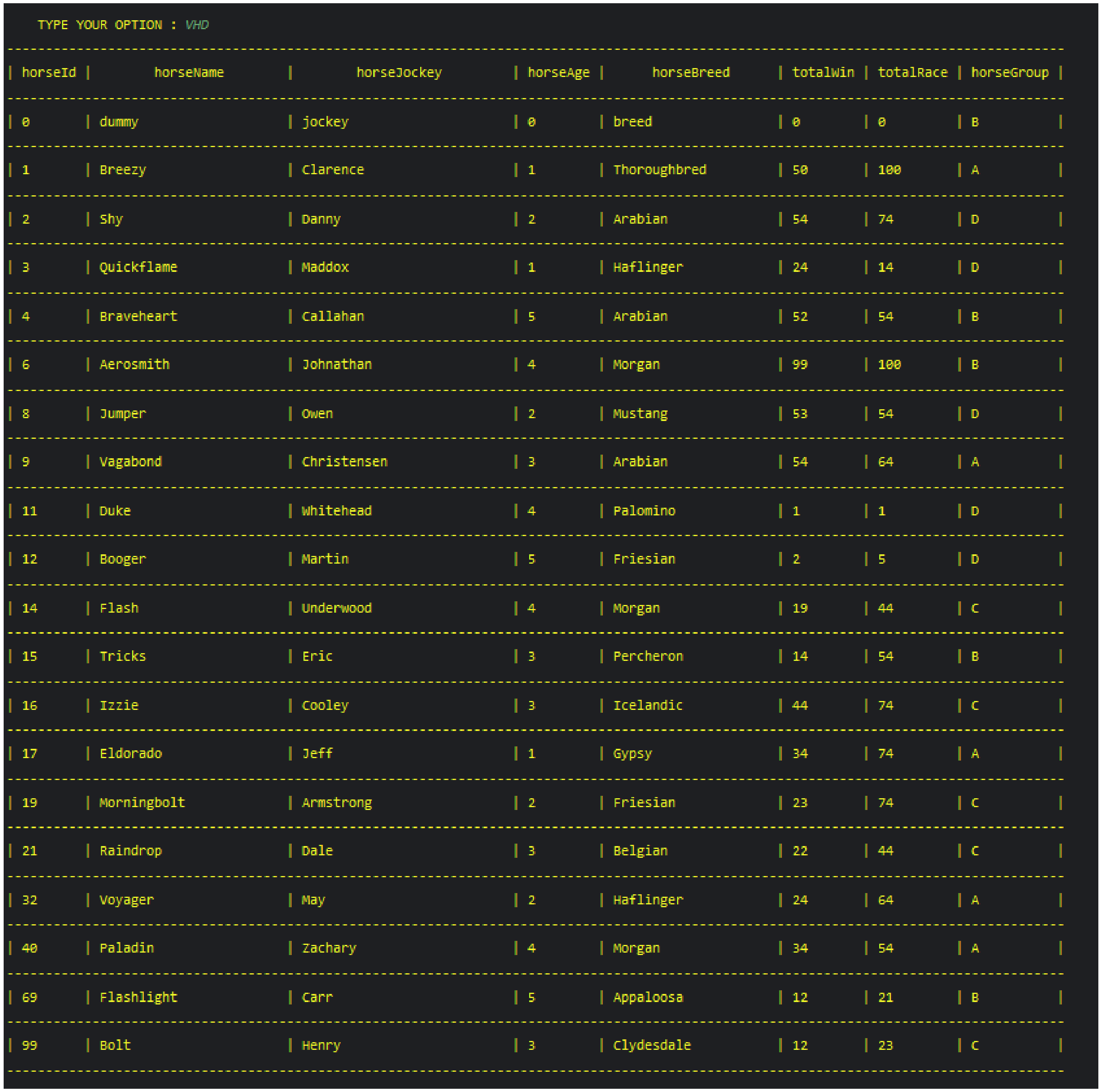
```
    TYPE YOUR OPTION : UHD
-   ENTER THE "Horse ID" OF THE HORSE'S RECORD TO UPDATE : 0
-  Type the Letter Corresponding to the word
-  I:ID | N:Name | J:Jockey | A:Age | B:Breed | R: TOTAL RACE | W: RACES WON | G: GROUP

- What do you what do update regrading this horse : J
Enter what you want to replace as: jockey
Horse ID : [1, 99, 11, 12, 14, 15, 32, 17, 16, 2, 4, 6, 9, 21, 19, 8, 3, 40, 69, 0]
Horse Name : ['Breezy', 'Bolt', 'Duke', 'Booger', 'Flash', 'Tricks', 'Voyager', 'Eldorado', 'Izzie', 'Shy', 'Braveheart', 'Aerosmith', 'Vagabond', 'Raindrop', 'Morningbolt', 'Jumper', 'Quickflame',
 'Paladin', 'Flashlight', 'dummy']
Horse Jockey : ['Clarence', 'Henry', 'Whitehead', 'Martin', 'Underwood', 'Eric', 'May', 'Jeff', 'Cooley', 'Danny', 'Callahan', 'Johnathan', 'Christensen', 'Dale', 'Armstrong', 'Owen', 'Maddox', 'Zachary
 ', 'Carr', 'jockey']
Horse Age : [1, 3, 4, 5, 4, 3, 2, 1, 3, 2, 5, 4, 3, 3, 2, 2, 1, 4, 5, 0]
Horse Breed : ['Thoroughbred', 'Clydesdale', 'Palomino', 'Friesian', 'Morgan', 'Percheron', 'Haflinger', 'Gypsy', 'Icelandic', 'Arabian', 'Arabian', 'Morgan', 'Arabian', 'Belgian', 'Friesian', 'Mustang',
 'Haflinger', 'Morgan', 'Appaloosa', 'breed']
Horse Win : [50, 12, 1, 2, 19, 14, 24, 34, 44, 54, 52, 99, 54, 22, 23, 53, 24, 34, 12, 0]
Horse Race : ['100', '23', '1', '5', '44', '54', '64', '74', '74', '74', '54', '100', '64', '44', '74', '54', '14', '54', '21', 0]
Horse Group : ['A', 'C', 'D', 'D', 'C', 'B', 'A', 'A', 'C', 'D', 'B', 'B', 'A', 'C', 'C', 'D', 'D', 'A', 'B', 'B']
[[0, 'dummy', 'jockey', 0, 'breed', 0, 0, 'B'], [1, 'Breezy', 'Clarence', 1, 'Thoroughbred', 50, '100', 'A'], [2, 'Shy', 'Danny', 2, 'Arabian', 54, '74', 'D'], [3, 'Quickflame', 'Maddox', 1, 'Haflinger',
 24, '14', 'D'], [4, 'Braveheart', 'Callahan', 5, 'Arabian', 52, '54', 'B'], [6, 'Aerosmith', 'Johnathan', 4, 'Morgan', 99, '100', 'B'], [8, 'Jumper', 'Owen', 2, 'Mustang', 53, '54', 'D'], [9,
 'Vagabond', 'Christensen', 3, 'Arabian', 54, '64', 'A'], [11, 'Duke', 'Whitehead', 4, 'Palomino', 1, '1', 'D'], [12, 'Booger', 'Martin', 5, 'Friesian', 2, '5', 'D'], [14, 'Flash', 'Underwood', 4,
 'Morgan', 19, '44', 'C'], [15, 'Tricks', 'Eric', 3, 'Percheron', 14, '54', 'B'], [16, 'Izzie', 'Cooley', 3, 'Icelandic', 44, '74', 'C'], [17, 'Eldorado', 'Jeff', 1, 'Gypsy', 34, '74', 'A'], [19,
 'Morningbolt', 'Armstrong', 2, 'Friesian', 23, '74', 'C'], [21, 'Raindrop', 'Dale', 3, 'Belgian', 22, '44', 'C'], [32, 'Voyager', 'May', 2, 'Haflinger', 24, '64', 'A'], [40, 'Paladin', 'Zachary ', 4,
 'Morgan', 34, '54', 'A'], [69, 'Flashlight', 'Carr', 5, 'Appaloosa', 12, '21', 'B'], [99, 'Bolt', 'Henry', 3, 'Clydesdale', 12, '23', 'C']]
Save Successful
```

DHD



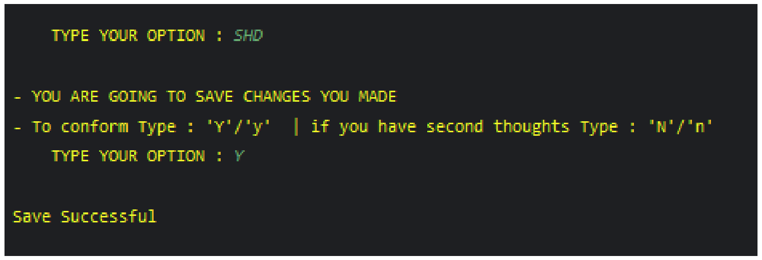
```
    TYPE YOUR OPTION : DHD
-   ENTER THE "Horse ID" OF THE HORSE'S RECORD TO DELETE : 0
[['0', 'Starsky', 'Mason', '2', 'Arabian', '50', '75', 'B'], ['2', 'Shy', 'Danny', '2', 'Arabian', '54', '74', 'D'], ['3', 'Quickflame', 'Maddox', '1', 'Haflinger', '24', '14', 'D'], ['4',
 'Braveheart', 'Callahan', '5', 'Arabian', '52', '54', 'B'], ['6', 'Aerosmith', 'Johnathan', '4', 'Morgan', '99', '100', 'B'], ['8', 'Jumper', 'Owen', '2', 'Mustang', '53', '54', 'D'], ['9',
 'Vagabond', 'Christensen', '3', 'Arabian', '54', '64', 'A'], ['11', 'Duke', 'Whitehead', '4', 'Palomino', '1', '1', 'D'], ['12', 'Booger', 'Martin', '5', 'Friesian', '2', '5', 'D'], ['14', 'Flash',
 'Underwood', '4', 'Morgan', '19', '44', 'C'], ['15', 'Tricks', 'Eric', '3', 'Percheron', '14', '54', 'B'], ['16', 'Izzie', 'Cooley', '3', 'Icelandic', '44', '74', 'C'], ['17', 'Eldorado', 'Jeff', '1',
  'Gypsy', '34', '74', 'A'], ['19', 'Morningbolt', 'Armstrong', '2', 'Friesian', '23', '74', 'C'], ['21', 'Raindrop', 'Dale', '3', 'Belgian', '22', '44', 'C'], ['32', 'Voyager', 'May', '2',
 'Haflinger', '24', '64', 'A'], ['40', 'Paladin', 'Zachary ', '4', 'Morgan', '34', '54', 'A'], ['69', 'Flashlight', 'Carr', '5', 'Appaloosa', '12', '21', 'B'], ['99', 'Bolt', 'Henry', '3',
 'Clydesdale', '12', '23', 'C']]
Save Successful
```

VHD



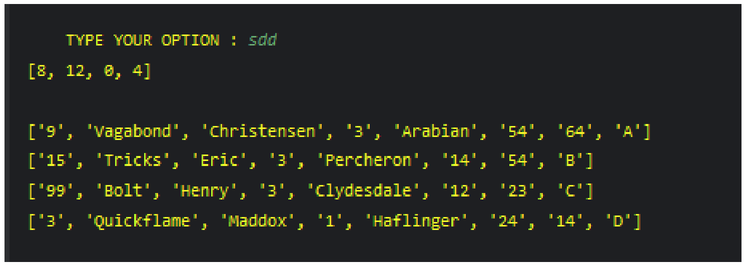
```
  TYPE YOUR OPTION : VHD
-------------------------------------------------------------------------------------------------------------------------------------
| horseId |        horseName         |         horseJockey         | horseAge |      horseBreed      | totalWin | totalRace | horseGroup |
-------------------------------------------------------------------------------------------------------------------------------------
| 0       | dummy                    | jockey                      | 0        | breed                | 0        | 0         | B          |
-------------------------------------------------------------------------------------------------------------------------------------
| 1       | Breezy                   | Clarence                    | 1        | Thoroughbred         | 50       | 100       | A          |
-------------------------------------------------------------------------------------------------------------------------------------
| 2       | Shy                      | Danny                       | 2        | Arabian              | 54       | 74        | D          |
-------------------------------------------------------------------------------------------------------------------------------------
| 3       | Quickflame               | Maddox                      | 1        | Haflinger            | 24       | 14        | D          |
-------------------------------------------------------------------------------------------------------------------------------------
| 4       | Braveheart               | Callahan                    | 5        | Arabian              | 52       | 54        | B          |
-------------------------------------------------------------------------------------------------------------------------------------
| 6       | Aerosmith                | Johnathan                   | 4        | Morgan               | 99       | 100       | B          |
-------------------------------------------------------------------------------------------------------------------------------------
| 8       | Jumper                   | Owen                        | 2        | Mustang              | 53       | 54        | D          |
-------------------------------------------------------------------------------------------------------------------------------------
| 9       | Vagabond                 | Christensen                 | 3        | Arabian              | 54       | 64        | A          |
-------------------------------------------------------------------------------------------------------------------------------------
| 11      | Duke                     | Whitehead                   | 4        | Palomino             | 1        | 1         | D          |
-------------------------------------------------------------------------------------------------------------------------------------
| 12      | Booger                   | Martin                      | 5        | Friesian             | 2        | 5         | D          |
-------------------------------------------------------------------------------------------------------------------------------------
| 14      | Flash                    | Underwood                   | 4        | Morgan               | 19       | 44        | C          |
-------------------------------------------------------------------------------------------------------------------------------------
| 15      | Tricks                   | Eric                        | 3        | Percheron            | 14       | 54        | B          |
-------------------------------------------------------------------------------------------------------------------------------------
| 16      | Izzie                    | Cooley                      | 3        | Icelandic            | 44       | 74        | C          |
-------------------------------------------------------------------------------------------------------------------------------------
| 17      | Eldorado                 | Jeff                        | 1        | Gypsy                | 34       | 74        | A          |
-------------------------------------------------------------------------------------------------------------------------------------
| 19      | Morningbolt              | Armstrong                   | 2        | Friesian             | 23       | 74        | C          |
-------------------------------------------------------------------------------------------------------------------------------------
| 21      | Raindrop                 | Dale                        | 3        | Belgian              | 22       | 44        | C          |
-------------------------------------------------------------------------------------------------------------------------------------
| 32      | Voyager                  | May                         | 2        | Haflinger            | 24       | 64        | A          |
-------------------------------------------------------------------------------------------------------------------------------------
| 40      | Paladin                  | Zachary                     | 4        | Morgan               | 34       | 54        | A          |
-------------------------------------------------------------------------------------------------------------------------------------
| 69      | Flashlight               | Carr                        | 5        | Appaloosa            | 12       | 21        | B          |
-------------------------------------------------------------------------------------------------------------------------------------
| 99      | Bolt                     | Henry                       | 3        | Clydesdale           | 12       | 23        | C          |
-------------------------------------------------------------------------------------------------------------------------------------
```

SHD



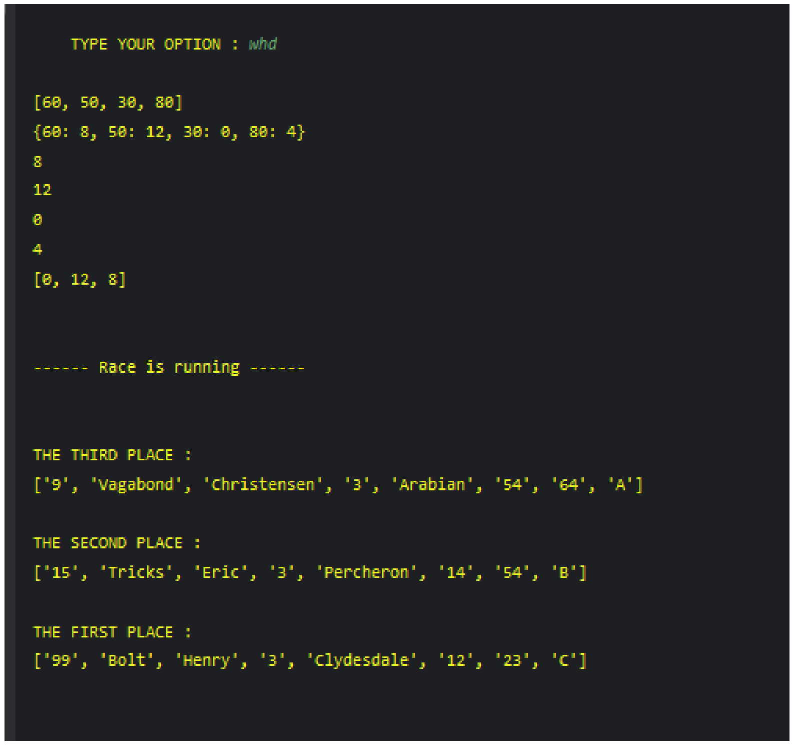
```
    TYPE YOUR OPTION : SHD

- YOU ARE GOING TO SAVE CHANGES YOU MADE
- To conform Type : 'Y'/'y'  | if you have second thoughts Type : 'N'/'n'
    TYPE YOUR OPTION : Y

Save Successful
```

SDD



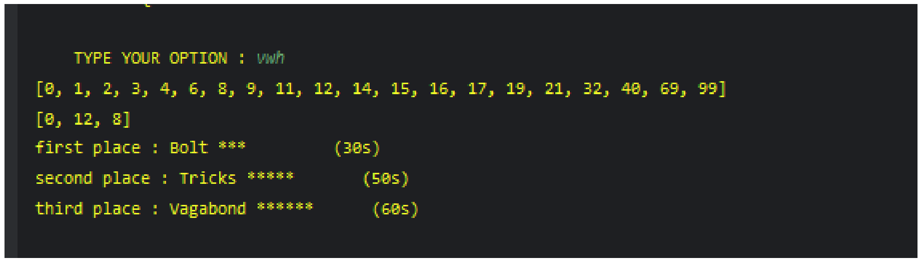
```
    TYPE YOUR OPTION : sdd
 [8, 12, 0, 4]

 ['9', 'Vagabond', 'Christensen', '3', 'Arabian', '54', '64', 'A']
 ['15', 'Tricks', 'Eric', '3', 'Percheron', '14', '54', 'B']
 ['99', 'Bolt', 'Henry', '3', 'Clydesdale', '12', '23', 'C']
 ['3', 'Quickflame', 'Maddox', '1', 'Haflinger', '24', '14', 'D']
```

WHD



```
    TYPE YOUR OPTION : whd

 [60, 50, 30, 80]
 {60: 8, 50: 12, 30: 0, 80: 4}
 8
 12
 0
 4
 [0, 12, 8]


 ------ Race is running ------



 THE THIRD PLACE :
 ['9', 'Vagabond', 'Christensen', '3', 'Arabian', '54', '64', 'A']

 THE SECOND PLACE :
 ['15', 'Tricks', 'Eric', '3', 'Percheron', '14', '54', 'B']

 THE FIRST PLACE :
 ['99', 'Bolt', 'Henry', '3', 'Clydesdale', '12', '23', 'C']
```

VWH



```
    TYPE YOUR OPTION : vwh
[0, 1, 2, 3, 4, 6, 8, 9, 11, 12, 14, 15, 16, 17, 19, 21, 32, 40, 69, 99]
[0, 12, 8]
first place : Bolt ***         (30s)
second place : Tricks *****       (50s)
third place : Vagabond ******      (60s)
```