Software Engineering and Management

DIT009 - Fundamentals of Programming

Assignment 1 - PokéNav App

You've been assigned to implement the software for the next generation of **PokéNav devices**. These should help Pokémon trainers in their journey, allowing them to do basic calculations and manipulate strings. Please follow the instructions below and fulfill each task to complete the assignment. **Details on submission and grading are provided in Canvas.** If you have any questions, contact your teachers.

Grading Criteria

For a grade 3 (pass) you must complete the points below:

- Your code must achieve **all functionalities** described in **Tasks 1 - 7**. You must follow the printing template provided in all output strings. We will use automated tests to check the templates.
- You must use the **suitable and correct programming constructs** to solve the tasks according to the specification.
- Deliver **readable, organized and understandable** code. This means proper choice of names, spacing and indentation, **using Python conventions**.

For a grade 4 (pass with merit):

- You must fulfill all criterias for a **grade 3**.
- You must complete **one of the** extra tasks. (8 or 9)

For a grade 5 (pass with distinction):

- You must fulfill all criterias for a **grade 3 and 4.**
- You must complete **both** extra tasks (8 and 9).

Learning Goals of the Assignment

The goal of this assignment is for you to **practice the basic concepts of procedural programming**. Here, we apply the following concepts:

- Printing
- Input
- Variables
- Conditionals
- Loops (for and while)
- Lists
- String manipulation
- Functions

Notes:

- You are allowed to use the `math` Python library. However, **any other library is not allowed in this assignment**.
- A general tip is to remember the string functions: `.upper(), .lower(), .title()`

Task 1: Creating the PokéNav Menu

In order to navigate the device, the user will use a *Main Menu* following the template below. Note that the menu should be the first thing to appear when you run the program.

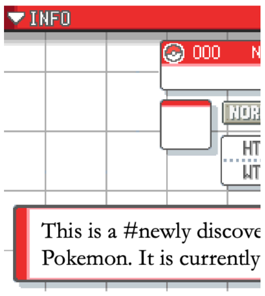
Typing the number of the option should take the user to the corresponding task described below. After completing the corresponding task, the user should return to the Main Menu. **The program must also end correctly, so `exit()` cannot be used.**

```
# OUTPUT
Welcome to the Main Menu. Choose one of the options below:

   1. Exit
   2. Identify hashtags
   3. Detect a palindrome
   4. Create an acronym
   5. Get Pokemon traits
   6. Match zodiac sign and element
   7. BMI calculator

Type your option:
```

When selecting option 1, present the following message to the user: `"Thank you for playing! See you next time!"`.

You can assume that the user will only input correct values - that is, integers between 1 and 7. If the group decides to later do the extra tasks, it is possible to add them in the menu following the correct order.

Task 2: Identify Hashtags



This is a #newly discove
Pokemon. It is currently

In the latest update, Pokéfinder, the Pokémon trainer social network, allowed users to add hashtags to their posts. Pokéfinder hashtags are used to define the popularity of any topic or trainer (e.g., training tips, upcoming battles, new Elite Four achievements, etc). Hashtags are an easy way to tag your content with the keyword you want to identify and highlight it with.

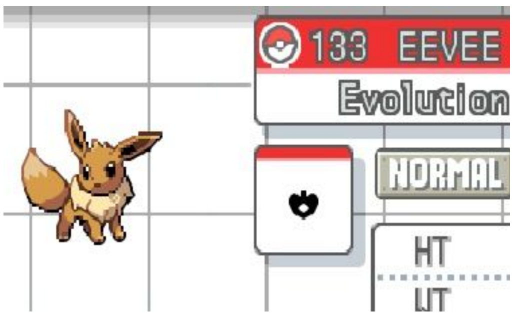
To help trainers identify the hashtags in a post, write a program that extracts and prints all words in a given sentence that start with the hashtag symbol (`#`). The user will input a single string. The program should print each hashtagged word on a new line and then return to the menu.

Notes:

- You can assume the user will type only valid strings (words and hashtags).
- Regular expressions (regex) **should not** be used to solve this task.
- Use the basic constructs covered in the learning goals to improve your basic skills.
- If the hashtag is repeated in the sentence, print it only once.

```
Example 1:
# INPUT
Type your post: "Today is a great day for #battle and #catch them
all. Love to #battle"
# OUTPUT
Hashtags found:
#battle
#catch

Example 2:
# INPUT
Type your post: "#hashtag1 is great but an even better one is
#hashtag2. #hashtag2"
# OUTPUT
Hashtags found:
#hashtag1
#hashtag2.
#hashtag2
```

Task 3: Detect a Palindrome



Recently, a new town that has a peculiar rule for their citizens has been built. In Poköping, all visiting trainers have to have their Pokémons' name be a palindrome i.e., a word that reads the same forwards and backwards.

Write a program that prompts the user to create a new Pokémon name. The program should output whether or not the name is a palindrome and then return to the menu.

You can assume the user will input a single word consisting only of letters (no spaces or special characters). **The program will also be case-insensitive** (e.g., "Madam" is considered a palindrome).

```
Example 1:
# INPUT
Type your Pokemon name: Eevee
# OUTPUT
The name 'Eevee' is a palindrome.

Example 2:
# INPUT
Type your Pokemon name: Charizard
# OUTPUT
The name 'Charizard' is not a palindrome.
```

Task 4: Create an Acronym

During battles, trainers need to quickly switch their lead Pokémon. However, typing the full name of some Pokémons in the PokéNav takes a long time. For these cases, the PokéNav wants to introduce a feature called *Name Acronyms*, in which longer names are abbreviated for ease of access.

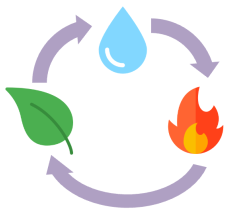
To do this, the PokéNav remembers a string composed of every n-th letter of the given Pokémon name. You are tasked with writing a program that helps the trainer figure out what the Name Acronym for a specific Pokémon is. The user provides the Pokémon name and the shortening factor (n) by which the name will be shortened. The program should output the **result on a new line, uppercase**, and then return to the menu.

Notes:

- You can assume the user will input a single word consisting of only letters.

```
Example 1:
# INPUT
Type your Pokemon name: Crabominable
Type your shortening factor: 4
# OUTPUT
Abbreviated name: BNE

Example 2:
# INPUT
Type your Pokemon name: Rattata
Type your shortening factor: 3
# OUTPUT
Abbreviated name: TT

Example 3:
# INPUT
Type your Pokemon name: Snorlax
Type your shortening factor: 8
# OUTPUT
Abbreviated name: # It will be empty
```

Task 5: Get Pokémon Traits



As a trainer, it is extremely important that you know your Pokémon's weaknesses and strengths. Write a program that helps trainers check a Pokémon's traits. The user will input the name and the type (Water, Fire or Grass) of a Pokémon, as a string. The program should output the information and then return to the main menu.

- **Fire-types** are weak against Water-types and strong against Grass-types
- **Water-types** are weak against Grass-types and strong against Fire-types
- **Grass-types** are weak against Fire-types and strong against Water-types

Notes:

- You can assume the user will input only valid strings (type names).
- The input is **case-insensitive** (e.g., "water" or "Water" should be considered the same).
- Only "Water", "Fire", and "Grass" are recognized as valid types.
- You don't need dictionaries to solve this task. Use only the constructs covered in the learning goals of the assignment.

```
Example 1:
# INPUT
Type your Pokemon name: Squirtle
Type your Pokemon type: Water
# OUTPUT
Squirtle is a Water-type Pokemon! It is strong against Fire-type
Pokemons and weak against Grass-type Pokemons.

Example 2:
# INPUT
Type your Pokemon name: Charmander
Type your Pokemon type: Fire
# OUTPUT
Charmander is a Fire-type Pokemon! It is strong against Grass-type
Pokemons and weak against Water-type Pokemons.
```

Task 6: Zodiac Sign and Eeveelution

In the PokéNav, the trainer profile includes your zodiac system, your element, and your Eeveelution (based on your element). Write a program that asks the user for their **birthday month index** and returns their **zodiac sign**, and corresponding **Eeveelution**.

For reference, these are the associations between the parts of the trainer profile above:

Index:	1	2	3	4	5	6	7	8	9	10	11	12
Month:	January	February	March	April	May	June	July	August	September	October	November	December
Zodiac Sign:	Capricorn	Aquarius	Pisces	Aries	Taurus	Gemini	Cancer	Leo	Virgo	Libra	Scorpio	Sagittarius

Zodiac sign	Element	Eeveelution
Cancer, Scorpio, Pisces	Water	Vaporeon
Aries, Leo, Sagittarius	Fire	Flareon
Taurus, Virgo, Capricorn	Earth	Leafeon
Gemini, Libra, Aquarius	Air	Jolteon

You can assume the user will type one of the twelve valid months. Print the sign, the element and the Eeveelution following the examples below.

```
Example:
# INPUT
Type your birth month: 7
# OUTPUT
Zodiac sign: Cancer
Element: Water
Eeveelution: Vaporeon
```

```
Example:
# INPUT
Type your birth month: 9
# OUTPUT
Zodiac sign: Virgo
Element: Earth
Eeveelution: Leafeon
```

Task 7: Implement Body Mass Index (BMI) Based on Height and Weight

Every Pokémon has a height and weight (in meters and kilos, respectively), and as trainers it's normal to want to keep our Pokémon happy and healthy. So, we want to add a program that calculates a Pokémon's BMI to better take care of them. Since there is no official metric for Pokémon BMI, we provide one for you:

- Underweight: x < 29
- Healthy: 29 <= x < 53
- Overweight: 53 <= x < 85
- Obese: x >= 85

To calculate BMI, you use height (in meters) and weight (in kilograms) in the following formula. You can assume the user will input valid positive numbers for height and weight.

$$BMI = \frac{weight}{height^2}$$

The program should print the BMI **rounded to two decimal places**, along with the **weight category**, then return to the main menu. If you would like some test cases with actual Pokémon, you can find all Pokémon BMI listed here. You can find our example on the next page.

```
Example 1:
# For height = 1.75 and weight = 68
# OUTPUT
BMI = 22.20. The Pokemon is underweight.

Example 2:
# For height = 1.2 and weight = 100
# OUTPUT
BMI = 69.44. The Pokemon is overweight.
```

Extra Task 8 : Fitness and Health Tracking

This year, the developers of PokéNav want to introduce a new, more capable version of their product, called **PokéNav PRO**. One of its premium features is fitness and health tracking for trainers and their Pokémons. They want you to implement a primitive version of the health app that for now, will only simulate tracking steps taken over 7 days. Ask the user to input the number of steps taken each day. The user should input 7 numbers separated by a comma. The program should then print the information below, and then return to the menu.

```
"Steps Statistics: <average_steps> + / - <std_steps> per day."
"Most active day: <week_day>. Least active day: <week_day>."
```

Where `<average_steps>` represents the average (mean) step count per day, `<std_steps>` represents the standard deviation of the population. Lastly, `<week_day>` represents the day of the week. To help you, the developers were kind enough to give you these math formulas:

$$\mu = \frac{\sum x_i}{N}$$

$$\mu = mean$$
$$\sum = sum\ of$$
$$x_i = each\ value\ from\ the\ population$$
$$N = size\ of\ population$$

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

$$\sigma = standard\ deviation$$
$$\sum = sum\ of$$
$$x_i = each\ value\ from\ the\ population$$
$$\mu = mean$$
$$N = size\ of\ population$$

The added functionality of the PokéNav PRO must be easily accessible from the menu, as with any other feature. For that, the main menu needs to be updated accordingly.

```
Welcome to the Main Menu. Choose one of the options below:

   1. Exit
   2. Identify hashtags
   3. Detect a palindrome
   4. Create an acronym
   5. Match zodiac sign and element
   6. Get Pokemon traits
   7. BMI calculator
   8. Fitness and health tracker

Type your option:
```

Notes:

- You can assume that the user will type positive integer numbers separated by comma.
- In case there are ties between the most and least active day, print the last day (see example cases below).
- The first number provided corresponds to Sunday, whereas the last will correspond to Saturday's steps.
- When reporting the average and standard deviation, the number should be automatically rounded to its closest two decimal places.
- Output weekdays have to be capitalized.

```
Example 1:
# INPUT
Step count per day: 3200, 1400, 5000, 2000, 100, 5000, 1740
# OUTPUT
Steps Statistics: 2634.29 + / - 1718.03 per day.
Most active day: Friday. Least active day: Thursday.

Example 2:
# INPUT
Step count per day: 1234, 1234, 1234, 1234, 1234, 1234, 1234
# OUTPUT
Steps Statistics: 1234.00 + / - 0.00 per day.
Most active day: Saturday. Least active day: Saturday.
```

Extra Task 9: Error-handling

The PokéNav should be the most reliable piece of technology that trainers own — it should never fail them during battle! Your task is to make sure that the PokéNav will be as robust as possible by implementing error-handling for every error-prone feature (task).

Error-handling for Task 1

Only options between 1 and 7 are acceptable. If the user enters a value out of the range, print an error message saying: “`Error - Invalid option. Please input a number between 1 and 7.`”. If you want to implement the extra functionality depicted in Extra Task 8, your message needs to update accordingly. Once the user provides a valid option, proceed with the tasks as per the specification.

```
Example:
# INPUT
Type your option: “11”
# OUTPUT (TASK 8 NOT IMPLEMENTED)
Error - Invalid option. Please input a number between 1 and 7.
# OUTPUT (TASK 8 IMPLEMENTED)
Error - Invalid option. Please input a number between 1 and 8.
```

Error-handling for Task 2

In case the user enters a sentence with no hashtags, print an error message containing the following content: `“No hashtags found.”`

```
Example:
# INPUT
Type your post: This sentence has no hashtags.
# OUTPUT
No hashtags found.
```

Error-handling for Task 5

If the type of the Pokémon is not valid, output the message `"Error - The Pokemon type provided is not valid. Valid types: Water, Fire, Grass."` and return to the menu.

```
Example:
# INPUT
Type your Pokemon name: Squirtle
Type your Pokemon type: apple
# OUTPUT
Error - The Pokemon type provided is not valid. Valid types:
Water, Fire, Grass.
```

Error-handling for Task 6

If the provided month index is not valid, present the following error message to the user: `"Error - The month index provided is not valid. Choose between 1 and 12."` and return to the menu.

```
Example:
# INPUT
Type your birth month: 13
# OUTPUT
Error - The month index provided is not valid. Choose between 1
and 12.
```

Error-handling for Task 7

A BMI calculator requires both the height and provided weight to be positive numbers. If not, display an error following the next template:

- If height was negative, but the weight was a positive integer, display: `"Error - Height must be a positive number."`
- If weight was negative, but the height was a positive integer, display: `"Error - Weight must be a positive number."`
- If both height and weight are not positive integers, display: `"Error - Height and weight must be positive numbers."`

After displaying the error message, return to the menu.

```
Examples:
# For height = 1.75 and weight = -2
# OUTPUT
Error - Weight must be a positive number.
```

```
# For height = -90 and weight = 5
# OUTPUT
Error - Height must be a positive number.

# For height = 0 and weight = -3
# OUTPUT
Error - Height and weight must be positive numbers.
```

Error-handling for Extra Task 8

Notes:

- If you are going for a grade 4 (pass with merit), we consider this task fulfilled without having Extra Task 8 error handling implemented.

If the number of days given does not equal a week-day, output the message `"Error - Invalid input. The program needs 7 numbers; you typed <n> numbers."` where `<n>` is to be replaced with the number of days provided, then return to the menu.

```
Example:
# INPUT
Step count per day: 100, 100, 200
# OUTPUT
Error - Invalid input. The program needs 7 numbers; you typed 3
numbers.
```

Extra Task (FOR THE GLORY!!!): Matrices

Notes:

- This task is **not graded** — it is here for you to discover and exercise a (maybe) new concept, called a matrix. Matrices in Python are achieved by creating a list of lists. Use online resources if you get stuck, but first try to think by yourself. Challenge yourself and remember, libraries are not needed to complete this exercise.

Inside any PokéNav, trainers store their team of Pokémons, each with three stats: Attack, Defense, and Speed. The device stores these stats and tracks level-ups and experience gained by each Pokémon. The device stores this data in a matrix where each row corresponds to a Pokémon, and each column corresponds to one of the three stats. For example:

```
stats = [
    [50, 60, 45],    # Pokemon 1 (Attack, Defense, Speed)
    [70, 65, 50],    # Pokemon 2 (Attack, Defense, Speed)
    [80, 85, 90],    # Pokemon 3 (Attack, Defense, Speed)
    [55, 50, 60]     # Pokemon 4 (Attack, Defense, Speed)
]
```

First, the PokéNav prompts the user to input the amount of Pokémons they want to store and their statistics.

```
Example:
# INPUT
Welcome to the Pokemon Stats Tracker Program!
How many Pokemons do you want to keep track of: "3"
Stats of your 1st Pokemon (Attack, Defense, Speed): "50, 60, 45"
Stats of your 2nd Pokemon (Attack, Defense, Speed): "70, 65, 50"
Stats of your 3rd Pokemon (Attack, Defense, Speed): "80, 85, 90"

# OUTPUT
Original Stats:
[50, 60, 45]
[70, 65, 50]
[80, 85, 90]
```

After each successful battle, the trainer's Pokémon stats are expected to increase by a fixed percentage, which can be applied as a scalar multiplier to the matrix. The PokéNav is expected to store the updated values and allow the system to further increase them following the same procedure. Here is an example of what that might look like:

```
Example:
# INPUT
Enter a multiplier to level up your Pokemons: 1.15

# OUTPUT
Updated Stats:
[57.5, 69.0, 51.75]
[80.5, 74.75, 57.5]
[92.0, 97.75, 103.5]
```

Your goal is to replicate this behavior in your PokéNav implementation. Go the extra mile for error-handling in this task. Good luck! 🙂