03-Exercises

October 28, 2024

1 03 - Exercises: loops

This week we saw Python's `for` and `while` loops. Here are the exercises to practice what we learnt :)

1.0.1 1. Animal Parade! - 4 points

Imagine you're organizing a parade with a list of animals. For each animal in the list, print its name multiple times, creating a "parade" effect based on the animal's length (number of characters). For instance, "Cat" (3 letters) will repeat 3 times.

Let's break down the task:

1. Create a list of animal names (e.g., `["Cat", "Elephant", "Dog", "Giraffe", "Bee"]`).
2. For each animal, print its name as many times as it has letters.
3. Use nested for loops to print the repeated names.

Sample Output:

```
Cat Cat Cat
Elephant Elephant Elephant Elephant Elephant Elephant Elephant Elephant
Dog Dog Dog
Giraffe Giraffe Giraffe Giraffe Giraffe Giraffe Giraffe
Bee Bee Bee
```

Choose 10 different animals :)

```
[ ]:
```

1.0.2 2. Aliens are landing…what are we gonna tell them?! - 4 points

An alien spaceship is counting down for takeoff, starting at 10 and going down to 0. Use a `while` loop to print each countdown number, but for each odd number, print "Odd - Hold Steady!", and for each even number, print "Even - Proceed".

Let's break down the task:

1. Set an initial countdown variable to 10.
2. Use a `while` loop to print each number and whether it's odd or even, as well as the corresponding message.
3. Stop the countdown when it reaches 0 and print "Blast Off!".

Sample Output:

```
10 - Even - Proceed
9 - Odd - Hold Steady!
8 - Even - Proceed
7 - Odd - Hold Steady!
6 - Even - Proceed
5 - Odd - Hold Steady!
4 - Even - Proceed
3 - Odd - Hold Steady!
2 - Even - Proceed
1 - Odd - Hold Steady!
Blast Off!
```

Hint: you might wanna pay special attention to *when* the while loop ends…

```
[ ]:
```

1.0.3 3. Vowel Counter with silly emojis - 4 points

Given a string, count how many vowels (a, e, i, o, u) it has. Print a different emoji for each vowel, like "A - , E - ," etc. (you can choose whatever you like, but the emoji's name has to start with that vowel!), and count each vowel type as you loop.

Let's break down the task:

1. Define a string with some text.
2. Use a `for` loop to iterate through each character in the string.
3. For each vowel found, print the corresponding emoji and increase a count.
4. Print the total number of each vowel found at the end.

Sample Output:

```
Text: "Hello, how are you today?"
E -
O -
O -
A -
E -
O -
U -
A -
Total A's: 2
Total E's: 2
Total O's: 3
Total U's: 1
```

Hint: there are several ways to incorporate emojis into code…ask Google, BIKI, or ChatGPT!

```
[ ]:
```

1.0.4 4. Mandatory grocery shopping - 4 points

You have two shopping lists, represented as **sets**, and you want to find:

- Items both lists have in common
- Items unique to each list

Let's break down the task:

1. Define two sets for the shopping lists: something like `list1 = {"apples", "bananas", "milk", "bread"}` and `list2 = {"milk", "bread", "eggs", "cheese"}`
2. Use **set operations** to find and print:

- Items in both lists
- Items unique to list1
- Items unique to list2

Sample output:

```
Items in both lists: {'milk', 'bread'}
Items unique to list1: {'apples', 'bananas'}
Items unique to list2: {'eggs', 'cheese'}
```

Hint: you might wanna ask Google, BIKI, ChatGPT or the Python Docs for which set operations are best to perform this task!

[ ]:

```
```

1.0.5 5. Internship at Uni Bielefeld Library this week - 4 points

You've been hired by Uni Bielefeld as the library hacker and you have a dictionary where each **book title is the key**, and the **value is a tuple** containing the total copies and copies currently checked out. Your task is to find the available books and sort them based on availability.

Let's break down the task:

1. Define a dictionary with **book titles as keys** and **tuples (total copies, checked-out copies) as values**:

```
library = {
"Python Programming": (10, 4),
"Data Science 101": (5, 1),
"Machine Learning": (7, 5),
"AI for Everyone": (6, 6),
"Algorithms Unlocked": (3, 1)
}
```

2. Calculate the available copies for each book and store these in a dictionary.
3. Print each book title along with its available copies, sorted by availability in descending order. Sample Output:

```
Available books sorted by availability:
Python Programming: 6 copies
Machine Learning: 2 copies
```

```
Data Science 101: 4 copies
Algorithms Unlocked: 2 copies
```

You can change the books' titles ;)

1.1 Bonus exercises

1.1.1 1. Magic square checker! - 1.5 points

A magic square is a square grid of numbers where the sums of each row, column, and diagonal are equal. Write a program to check if a given matrix is a magic square.

Let's break down the task:

1. Define a 3x3 or 4x4 matrix (you can start by choosing a known magic square like `[[8, 1, 6], [3, 5, 7], [4, 9, 2]]`).
2. Use `for` loops to calculate the sums of each row, column, and diagonal.
3. Check if all the sums are equal, and print "It's a magic square!" if true, or "Not a magic square."

Sample Output:

```
Matrix: [[8, 1, 6], [3, 5, 7], [4, 9, 2]]
Output: It's a magic square!
```

Hint: pay attention, lists are nested!

[ ]:

```

```

1.1.2 2. Forest Fire Simulation - 1.5 points

Unfortunately, a fire has broken down in Teutoburg forest. To save the forest we have to be quick and try to simulate how the fire spreads, so that we can contain it.

Imagine a simple grid of trees, represented by a 2D list, where each cell can be " " (tree) or " " (fire). Write a program to simulate how a fire spreads over several steps.

Let's break down the task:

1. Define a 5x5 grid with some trees (" ") and one fire (" ").
2. Use a `while` loop to simulate each time step of the fire spreading:
3. For each cell that's " ," turn adjacent " " cells into " ."
4. Print every step and stop when there are no more trees left to burn.

Example Grid Progression: `Initial:`

```
After Step 1:
```

...

```
[ ]:
```