L12 Tasks

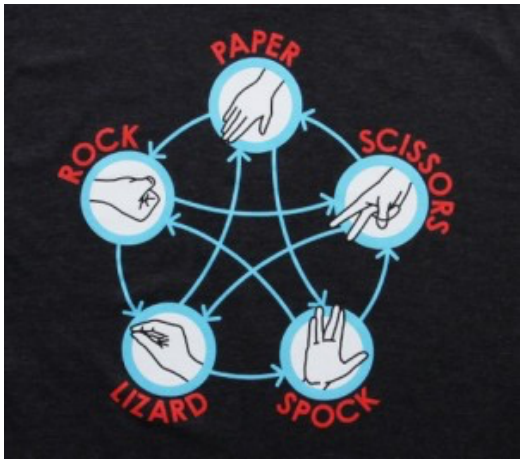
- Rock-Paper-Scissors-Lizard-Spock
- Getting data from Web – Rick & Morty characters
- Web UI SpaceX launch search of crew missions & their crew

L12-T1: Rock-Paper-Scissors-Lizard-Spock

NOTE: You should have all the required skills to plan & implement this task, even though the instructions are "a bit vague" :-)

The task of this exercise is to program an infamous game of Rock-Paper-Scissors-Lizard-Spock. For this task definition, check out the YouTube video: https://www.youtube.com/watch?v=x5Q6-wMx-K8

The Difference between this and regular rock-paper-scissors is a small added complexity, now each option can beat two different choices and can also lose to two different choices. Here is a diagram defining the relationships:



Create a game loop (that's just a regular while loop!) that asks for user input. User should select one of the options from Rock-Paper-Scissors-Lizard-Spock or type "exit" to stop the program. The game must play repeatedly until the player wants to exit.

Initialize the options as a list of strings ["Rock", "Paper",…]. Use Python's random-module to select the computer's answer from the list.

HINT: This is probably easiest to implement with multiple IF-ELSE structures.

Example run:

```
Welcome to Rock-Paper-Scissors-Lizard-Spock!
Choose Rock, Paper, Scissors, Lizard, or Spock (type 'exit' to quit): rock
The computer chose: Lizard
```

```
You won! Rock beats Lizard.
Do you want to play again? (yes/no): yes
Choose Rock, Paper, Scissors, Lizard, or Spock (type 'exit' to quit): spock
The computer chose: Lizard
You lost! Lizard beats Spock.
Do you want to play again? (yes/no): no
Thanks for playing! Goodbye!
```

L13-T2: Fetching data from the web – Rick & Morty characters

In this task we want to fetch data from public APIs on the internet. Our case is to analyze characters details in Rick & Morty series. The series is famous for being "rather violent" and containing parallel universes. To analyze the characters and the series, we are going to write couple of scripts to understand the characters a bit better.

In our project, we will use the public Rick & Morty API: https://rickandmortyapi.com/documentation/#character

The API provides a lot of data (hundreds of characters for example). Because of that, the API is paginated, e.g. one query only returns 20 results. Here we will learn to use "pages", e.g. requestion for the next 20 results.

Your goal is to do the following things:

1. Create a menu that asks for user input.
    a. User can choose to get all dead characters from the show. If user chooses this option, he/she is asked for how many pages to retrieve
    b. User can also choose to search for a specific character name and their living status (alive, dead, unknown). This input also asks for the maximum pages to retrieve
2. Once chosen, the code should request for the given data.
    a. If the data contains value ["info"]["next"], it points to the next page. If the user has requested for more than 1 page, we should also fetch this "next" page and keep going until we have the wanted amount of pages
    b. If there are less than the wanted amount of pages, just show the results that were found (not all searches return 10 pages!)
    c. In all cases, print the details of the found character. See the example run for details to print

HELPERS:

1. Use the following imports to get the data

```
from urllib.request import urlopen
import json
```

2. Use the following command to call & read the JSON results

```
with urlopen(url) as response:
        body = json.load(response)
```

3. To check for the next page, use the following command (or similar)

```
next_page = body["info"]["next"]
```

Example run

```
Welcome to Rick & Morty API
You have the following options
0) Exit
1) Get all dead characters
2) Get characters by their name and living status
Enter your selection: 1
Enter limit of pages: 3
Getting page 1 url: https://rickandmortyapi.com/api/character?status=dead
Getting page 2 url: https://rickandmortyapi.com/api/character?page=2&status=dead
Getting page 3 url: https://rickandmortyapi.com/api/character?page=3&status=dead
ID:8 - name:Adjudicator Rick - type: species:Human - origin:unknown - status:Dead
ID:9 - name:Agency Director - type: species:Human - origin:Earth (Replacement Dime
nsion) - status:Dead
ID:10 - name:Alan Rails - type:Superhuman (Ghost trains summoner) species:Human -
origin:unknown - status:Dead
ID:11 - name:Albert Einstein - type: species:Human - origin:Earth (C-137) - status
:Dead
ID:12 - name:Alexander - type: species:Human - origin:Earth (C-137) - status:Dead

…removed some entries for simplicity’s sake…

ID:141 - name:Ghost in a Jar - type:Parasite species:Alien - origin:unknown - stat
us:Dead
Total number of 60 dead characters
You have the following options
0) Exit
1) Get all dead characters
2) Get characters by their name and living status
Enter your selection: 2
Enter character to search for: morty
Enter living status (alive, dead, unknown): alive
Enter limit of pages: 3
Getting page 1 url: https://rickandmortyapi.com/api/character?name=morty&status=al
ive
Getting page 2 url: https://rickandmortyapi.com/api/character?page=2&name=morty&st
atus=alive
ID:2 - name:Morty Smith - type: species:Human - origin:unknown - status:Alive
ID:18 - name:Antenna Morty - type:Human with antennae species:Human - origin:unkno
wn - status:Alive
ID:27 - name:Artist Morty - type: species:Human - origin:unknown - status:Alive

..removed some results for simpicity’s sake…

ID:759 - name:Turkey Morty - type:Turkey species:Animal - origin:Earth (Replacemen
t Dimension) - status:Alive
ID:805 - name:Baby Mouse Skin Morty - type: species:Human - origin:Citadel of Rick
s - status:Alive
Total number of 34 alive morty's found
You have the following options
0) Exit
1) Get all dead characters
2) Get characters by their name and living status
Enter your selection: 0
```

L12-T3: Displaying SpaceX Crewed Mission Details

SpaceX – the space company is publishing their launch details via public API. Our goal is to create a search system that fetches details of manned missions during a given year and gives honor to the men & women who conquer space one reusable rocket at a time :-P

NOTE: We have not learned how to actually create HTML and CSS files and styles. This task contains some ready-made .html pages and a .css file. You can use these to your advantage. In this exercise, AI will help populate the webpage contents if you struggle with this. Just make sure to give clear prompts, or it will break your pages.

For this task, we will be using SpaceX's Launches V5 and Crew V4 APIs to get details about them. See the API descriptions here:

- https://github.com/r-spacex/SpaceX-API/blob/master/docs/launches/v5/one.md
- https://github.com/r-spacex/SpaceX-API/blob/master/docs/crew/v4/one.md

You will need to use the Python's flask library to create the web page. You can do it by running *pip install flask*.

In order to run the flask app, you need to call start it as follows:

```
if __name__ == "__main__":
    app.run(debug=True)
```

As the code is rather complex and especially the web UI requires knowledge of many new concepts such as HTML and CSS, some of the code is already predefined. Your task is to fill the missing parts.

app.py

This is the main program that acts as the server that returns data. This one is started when you run the flask program.

Fill in the following functions:

def fetch_spacex_data()

- Get the data from the launches API
- Filter out all missions where there is no crew present

def mission(mission_id)

- Find the mission details of given mission.
- Fetch all the launches
- Try to find the one that matches with this mission_id and return it as mission_details

def crew(crew_id)

- Fetch all the crew members

- Try to find the crew_id member from all the crew members. Return the member as crew_member

mission.html

This page displays the details of the mission.

It is missing the links to Crew Members and the media details. Populate these with the needed UI components.

crew.html

This is the website that shows information related to the crew members.

Try to edit this yourself to get a nice looking crew member website to display.

You can take a look example from mission.html and use for example ChatGPT to help you with this crew.html file.