Selected dataset: Wiki 40B, https://huggingface.co/datasets/google/wiki40b

We selected this dataset due to its diverse corpus of languages. The primary set of languages we train on is English, Russian, and Chinese. These three languages correspond to the nations with the largest space programs.
We next select countries with primary languages not represented within the primary set of languages, and that send a great number of visitors to the ISS. This includes Japanese, Italian, French, and German, as well as top languages per population such as Hindi, Spanish, Portuguese, and Arabic.
Finally, we train on other languages to a smaller amount. Visualized, after cleaning Wiki40B's data to match our use case, we aim to have the training languages approximately broken down like such:

Languages	Proportion
English, Russian, Chinese	70%
Japanese, Italian, French, German, Hindi, Spanish, Portuguese, Arabic	27.5%
All other languages[1]	2.5%

However, this dataset includes code words like `_START_ARTICLE_` or `_START_PARAGRAPH_`, which is not something we want our n-gram to train on as a real astronaut will not use this sort of language. As a result, we plan to pre-process this Hugging Face dataset ourselves and publish a new dataset with these words omitted or replaced with a `\n`. However we will not do this, as well as training language distribution, in Checkpoint 1 as this will take some work and we are deciding to focus more on the core functionality of the program.

Method

N-gram, optimized through the use of tries and backoff mechanism.

Methodology Outline:
Predicts next Unicode character given context provided by previous choices of astronaut using N-gram on Unicode characters. In the future, backoff and Kneser-Ney smoothing will be implemented to support the N-gram model. This n-gram language model looks at contexts of various sizes (current maximum of 7 characters), using tries to store context prefixes and count of next possible characters at each node. This is then used to compute probabilities for all observed characters. Then, using those probabilities, we return the top 3 most likely characters for the astronaut to choose from.

However, we note that we have not implemented Kneser-Ney or backoff, as in Checkpoint 1 our focus was a program that follows the spec, as opposed to a fully fleshed implementation. This mirrors the instructions focus on producing a program that follows the spec, not requiring error

[1]On edstem, it's mentioned that any human language is possible.

or processing speed measurements which the backoff and Kneser-Ney smoothing would have assisted with.

Implementation Details:

- Language: Python 3
- Frameworks + Libraries:
    - Pickle - saving model state
    - Argparse - parsing arguments
    - Datasets - loading HuggingFace datasets

[1]On edstem, it's mentioned that any human language is possible.