Author: Jay Jitesh Chavan
Python Programmer

Phone Number Analysis Using Python

Introduction

This report explains the functionality of a Python program designed to analyze phone numbers using the `phonenumbers` library. The code takes a phone number as input, verifies its validity, and provides details such as the phone number type, carrier, timezone, and region. It also logs the analysis results to a file for future reference.

Code Overview

The script consists of several key components:

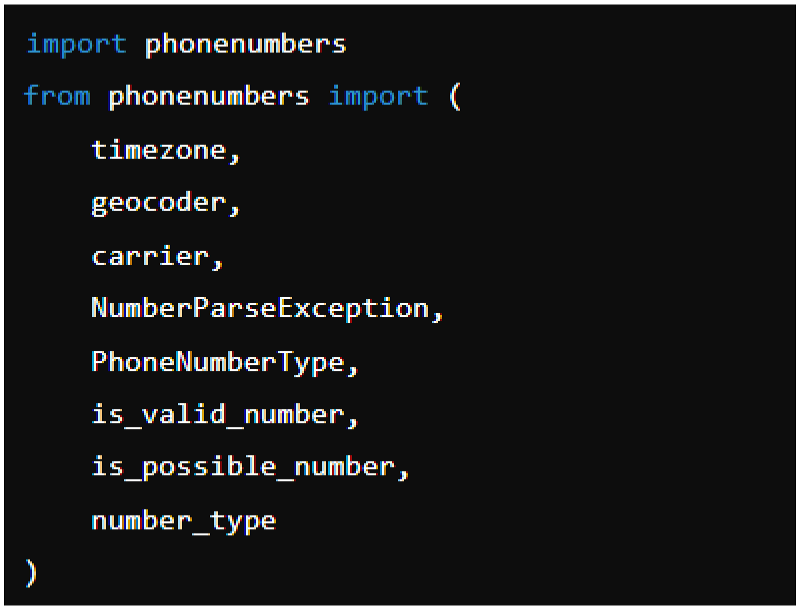
- **Library Imports:** The `phonenumbers` library provides the functionality to parse, validate, and analyze phone numbers. This includes extracting information such as the carrier, geographical region, and timezone associated with a phone number.

- **Functions and Methods:**
  - **“Parse”:** Used to parse the input number into a structured format.
  - **“is_possible_number” and “is_valid_number”:** Check whether the input phone number is possible and valid.
  - **“number_type”:** Determines the type of phone number, such as mobile, landline, or VoIP.
  - **“timezone.time_zones_for_number”,“carrier.name_for_number”, and “geocoder.description_for_number”:** Extract additional information like time zone, carrier, and geographical region.

- **Error Handling:** The program catches errors related to phone number parsing (`NumberParseException`) and general exceptions to prevent the script from crashing unexpectedly.

Detailed Code Breakdown

Library Imports



```
import phonenumbers
from phonenumbers import (
    timezone,
    geocoder,
    carrier,
    NumberParseException,
    PhoneNumberType,
    is_valid_number,
    is_possible_number,
    number_type
)
```

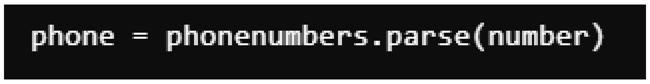
The necessary modules are imported from the phonenumbers library:

- **Timezone :** For retrieving the time zones linked to the phone number.
- **Geocoder :** To obtain the geographical region.
- **Carrier :** To get the mobile carrier's name.
- **NumberParseException :** For catching parsing errors.
- **PhoneNumberType :** Enum values for different types of phone numbers.
- **is_valid_number and is_possible_number :** To verify number validity.
- **number_type :** To classify the phone number as mobile, landline, etc.

Analyze Phone Number Function

The core of the script lies in the `analyze_phone_number` function. This function takes a phone number as input, parses it, and performs multiple checks and analyses.

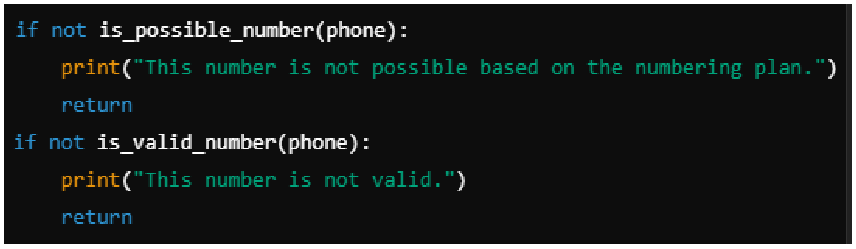
1. Phone Number Parsing:



```
phone = phonenumbers.parse(number)
```

The `parse` function converts the user input into a phone number object. If the input is improperly formatted or invalid, an exception is raised.

2. Validity Checks:

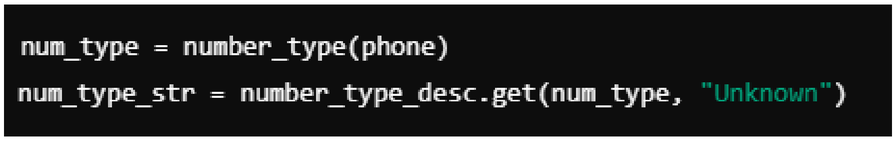


```python
if not is_possible_number(phone):
    print("This number is not possible based on the numbering plan.")
    return
if not is_valid_number(phone):
    print("This number is not valid.")
    return
```

Two checks are performed:

- **is_possible_number:** Verifies if the number adheres to the numbering plan of the country.
- **is_valid_number:** Ensures the phone number is valid based on country-specific rules.

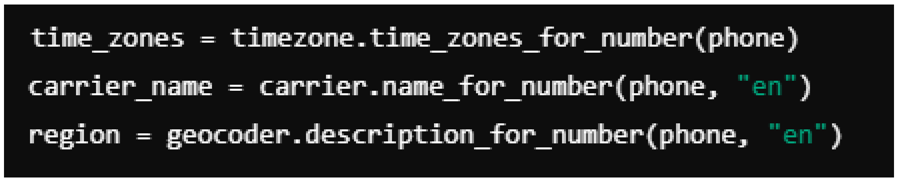
3. Phone Number Type:



```python
num_type = number_type(phone)
num_type_str = number_type_desc.get(num_type, "Unknown")
```

The type of phone number is determined (e.g., mobile, landline, VoIP). The script uses a dictionary (`number_type_desc`) to map numeric codes to human-readable descriptions.

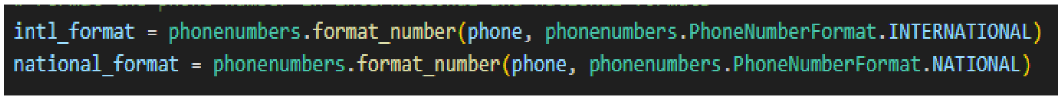
4. Extracting Timezone, Carrier, and Region:



```python
time_zones = timezone.time_zones_for_number(phone)
carrier_name = carrier.name_for_number(phone, "en")
region = geocoder.description_for_number(phone, "en")
```

- **time_zones_for_number:** Provides the time zones associated with the phone number.
- **name_for_number:** Retrieves the carrier’s name (e.g., Verizon, AT&T).
- **description_for_number:** Identifies the geographical region based on the number.

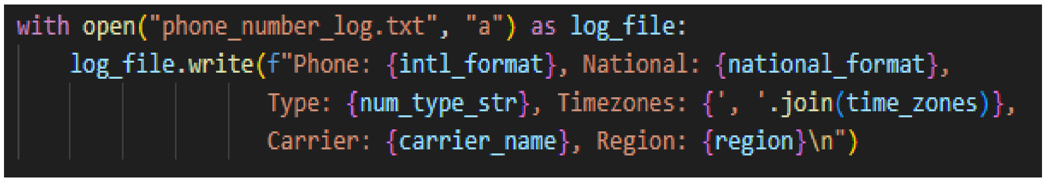
5. Formatting and Output:



```python
intl_format = phonenumbers.format_number(phone, phonenumbers.PhoneNumberFormat.INTERNATIONAL)
national_format = phonenumbers.format_number(phone, phonenumbers.PhoneNumberFormat.NATIONAL)
```

The phone number is formatted in both international and national formats. This ensures clarity and consistency in the output.

6. Log Results:

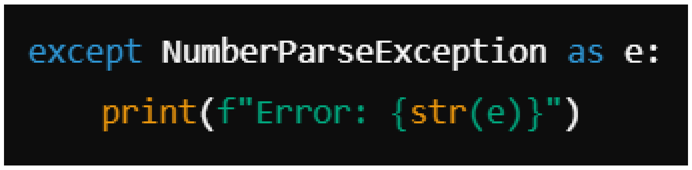


```python
with open("phone_number_log.txt", "a") as log_file:
    log_file.write(f"Phone: {intl_format}, National: {national_format},
                    Type: {num_type_str}, Timezones: {', '.join(time_zones)},
                    Carrier: {carrier_name}, Region: {region}\n")
```

The results of the analysis are logged in a file (`phone_number_log.txt`) for future reference. This helps in keeping a record of analyzed numbers.

Error Handling

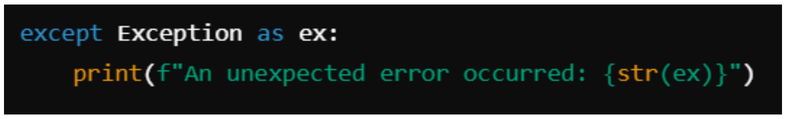
- **Phone Number Parsing Errors:**



```python
except NumberParseException as e:
    print(f"Error: {str(e)}")
```

Any issues during phone number parsing are captured and displayed to the user.

General Errors:

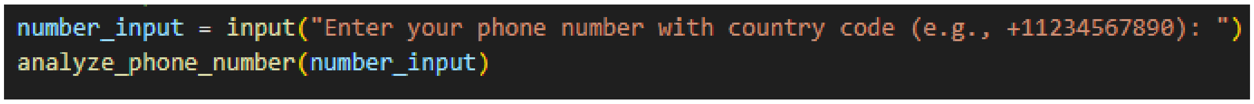


```
except Exception as ex:
    print(f"An unexpected error occurred: {str(ex)}")
```

Other unexpected errors are caught to prevent the program from crashing, ensuring robustness.

User Input

The program takes user input:

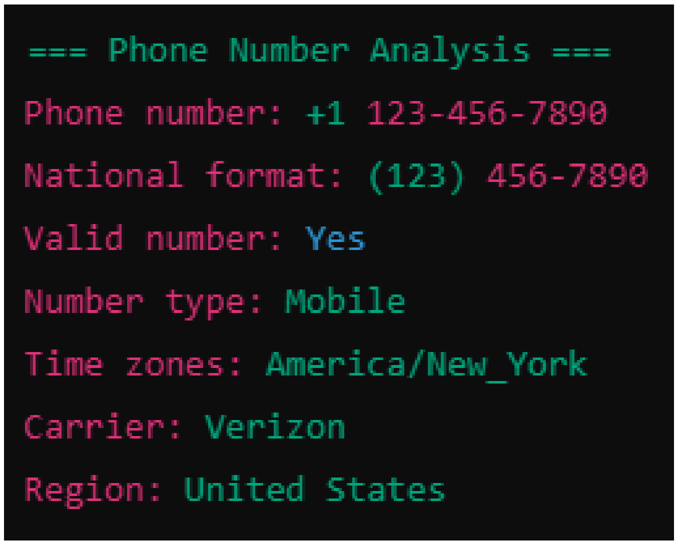


```
number_input = input("Enter your phone number with country code (e.g., +11234567890): ")
analyze_phone_number(number_input)
```

The input format requires the country code (e.g., +1 for the U.S.).

Example Output

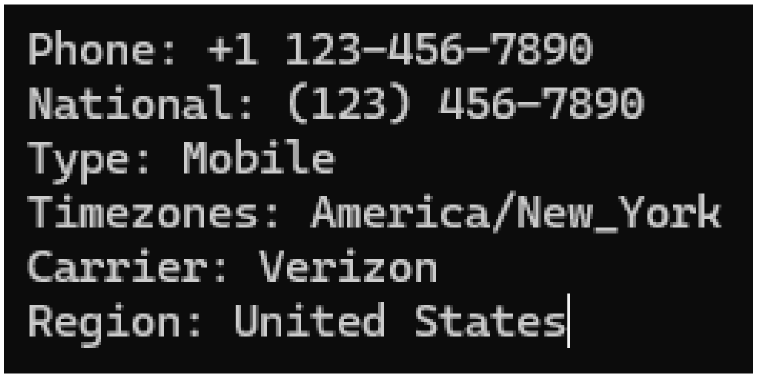
Here is an example of what the output might look like after analyzing a valid phone number:



```
=== Phone Number Analysis ===
Phone number: +1 123-456-7890
National format: (123) 456-7890
Valid number: Yes
Number type: Mobile
Time zones: America/New_York
Carrier: Verizon
Region: United States
```

Log Entry Example

The following is a sample entry written to `phone_number_log.txt`:



```
Phone: +1 123-456-7890
National: (123) 456-7890
Type: Mobile
Timezones: America/New_York
Carrier: Verizon
Region: United States
```

Conclusion

This phone number analysis script provides a robust and flexible solution for verifying and analyzing phone numbers. It ensures that input numbers are valid and provides detailed insights like phone number type, carrier, time zones, and geographical region. Additionally, it handles exceptions gracefully and logs results to a file for record-keeping.