Validation.py

Release 1.0.0

Deeapk Raj

Dec 12, 2022

CONTENTS

Documentation for the Morse code library is in the works. it will be available soon.

Contribute to Morse documentation on GitHub check out the source code for this documentation.

CHAPTER

ONE

INTRODUCTION

This project is python port of Validator.js which is a library for string validation. So people who are familiar with Validator.js can easily switch to this library. It is also a good way to learn how to port a library from one language to another. I have tried to keep the code as similar as possible to the original library. if you find any bugs or have any suggestions please open an issue. if you want to contribute please open a pull request.

CHAPTER

TWO

DOCUMENT VALIDATION

2.1 Aadhar Card Validation

aadhar card is 12 digit number. *Vorhoef Algorithm* is use to validate aadhar card number.

```
from sanatio import Validator

val = Validator()
```

isAadharCard(value) - check if the value is a valid aadhar card number.

```
>>> val.isAadharCard('9284 9436 2499')
True
```

2.2 Card Validation function

This function is used to validate the card number and the card type. *Luhn algorithm* is used to validate the card number. The function returns boolean value.

```
from sanatio import Validator

val = Validator()
```

isCreditCard(value) - Checks if the value is a valid credit card number.

```
>>> val.isCreditCard('5191914942157165')
True
>>> val.isCreditCard('5191914942157166')
True
```

2.3 License Plate Validations

Check if a license plate is valid or not.

```
from sanatio import Validator

val = Validator()
```

isLicensePlate() - Check if a license plate is valid or not.

args: value

```
>>> val.isLicensePlate('UP12AB1234')
True
```

2.4 Mobile Number Validation

This is a simple library to validate mobile numbers. It is based on the list of mobile number ranges published by the ITU-T E.164. The list is updated

```
from sanatio import Validator

val = Validator()
```

`isMobilePhone(value, locale)` - checks if the string is a valid mobile phone number.

```
>>> val.isMobilePhone('987654321', 'IN')
```

2.5 Pan Card Validation

2.6 Passport Number Validation

This module provides a validation for passport numbers.

```
from sanatio import Validator

val = Validator()
```

isPassportNumber(value, locale): - Return True if the string is a valid passport number.

Args: value, locale

```
>>> val.isPassportNumber('A1234567', "IN")
True
```

2.7 Postal Code Validation Functions

The following functions are used to validate postal codes.

```
from sanatio import Validator

val = Validator()
```

isPostalCode(value, locale) - return true if the postal code is valid for the country

args: value, locale

```
>>> val.isPostalCode(value='110016', locale='IN')
True
```

```
>>> val.isPostalCode(value='10133-1234', locale='US')
True
```

CHAPTER

THREE

STRING VALIDATION

3.1 String Validation functions

The following functions are used to validate strings.

```
from sanatio import Validator

val = Validator()
```

equals(value1, value2, ignoreCase)
Returns true if the two strings are equal.

```
>>> val.equals("abc", "abc")
True
>>> val.equals("abc", "ABC")
False
>>> val.equals("abc", "ABC", ignoreCase=True)
True
```

isLength(value, min, max)
Returns true if the string is between the specified min and max.

```
>>> val.isLength("abc", 2, 3)
True
>>> val.isLength("abc", 2, 2)
False
```

isEmpty(value)
Returns true if the string is empty.

```
>>> val.isEmpty("")
True
>>> val.isEmpty("abc")
False
```

isAlphanumeric(value)
Returns true if the string is alphanumeric.

```
>>> val.isAlphanumeric("abc123")
True
>>> val.isAlphanumeric("abc123!")
False
```

isAlpha(value)

Returns true if the string is alphabetic.

```
>>> val.isAlpha("abc")
True
>>> val.isAlpha("abc123")
False
```

contains(value, substring)

Returns true if the string contains the substring.

```
>>> val.contains("abc", "a")
True
>>> val.contains("abc", "d")
False
```

CHAPTER

FOUR

NUMBER VALIDATION

4.1 Number Validation functions

The following functions are used to validate numbers.

```
from sanatio import Validator

val = Validator()
```

isDecimal(value)

Returns true if the value is a decimal number.

```
>>> val.isDecimal(1)
True
>>> val.isDecimal(1.0)
True
```

isDivisibleBy(value, divisor)

Returns true if the value is divisible by the divisor.

```
>>> val.isDivisibleBy(10, 2)
True
>>> val.isDivisibleBy(10, 3)
False
```

CHAPTER

FIVE

DATE VALIDATION

5.1 Date Validation Functions

The following functions are used to validate dates. They return a boolean value.

```
from sanatio import Validator

val = Validator()
```

`IsDate()` Returns true if the value is a valid date.

args: value

```
>>> val.is_date('2012-12-12')
True
```

`isBefore()` Returns true if the value is before the date.

args: date1, date2

date2 is optional, if not provided, the current date is used.

```
>>> val.is_before('2012-12-12', '2012-12-13')
True
```

`isAfter()` Returns true if the value is after the date.

args: date1, date2

date2 is optional, if not provided, the current date is used.

```
>>> val.is_after('2012-12-12', '2012-12-11')
True
```

CHAPTER

SIX

STRING/NUMBER CONVERSION

CHAPTER

SEVEN

USERNAME VALIDATION

7.1 Username Validation

The following validation rules are applied to the username:

```
from sanatio import Validator

val = Validator()
```

isDiscordUsername(value) - Checks if the username is a valid Discord username.

```
>>> val.isDiscordUsername('test#1234')
True
```

CHAPTER

EIGHT

PASSWORD VALIDATION

8.1 Password Validation

The password validation is done by the function isStrongPassword().

```
from sanatio import Validator

val = Validator()
```

isStrongPassword(value)

Returns true if the password is strong enough, false otherwise.

```
>>> val.isStrongPassword('123456')
False
>>> val.isStrongPassword('123456789@Abc')
True
```

CHAPTER

NINE

SUPPORTED COUNTRY

9.1 Supported Country for MobileNumber

Country	Document Type
India	
United States	
Afghanistan	
Saudi Arabia	
Singapore	
South Africa	
Pakistan	
Armenia	
Argentina	
Austria	
Australia	
Azerbaijan	
Bangladesh	
Belarus	
Brazil	
Canada	
Switzerland	
China	
Cyprus	
Czech Republic	
Germany	
Denmark	
Algeria	
Spain	
Finland	
France	
United Kingdom	
Ireland	
Indonesia	
Iran	
Italy	
Japan	
Libya	
Malaysia	

continues on next page

Table 1 – continued from previous page

Country	Document Type
Mexico	
Poland	
Portugal	
Sweden	
Thailand	
Turkey	
Ukraine	

9.2 Supported Country for PassportNumber

Country	Document Type
India	
United States	
Afghanistan	
Saudi Arabia	
Singapore	
South Africa	
Pakistan	
Armenia	
Argentina	
Austria	
Australia	
Azerbaijan	
Bangladesh	
Belarus	
Brazil	
Canada	
Switzerland	
China	
Cyprus	
Czech Republic	
Germany	
Denmark	
Algeria	
Spain	
Finland	
France	
United Kingdom	
Ireland	
Indonesia	
Iran	
Italy	
Japan	
Libya	
Malaysia	
Mexico	
Poland	

continues on next page

Table 2 – continued from previous page

Country	Document Type
Portugal	
Sweden	
Thailand	
Turkey	
Ukraine	

9.3 Supported Country for PostalCode

Country	Document Type
India	
United States	
Afghanistan	
Saudi Arabia	
Singapore	
South Africa	
Pakistan	
Armenia	
Argentina	
Austria	
Australia	
Azerbaijan	
Bangladesh	
Belarus	
Brazil	
Canada	
Switzerland	
China	
Cyprus	
Czech Republic	
Germany	
Denmark	
Algeria	
Spain	
Finland	
France	
United Kingdom	
Ireland	
Indonesia	
Iran	
Italy	
Japan	
Libya	
Malaysia	
Mexico	
Poland	
Portugal	
Sweden	

continues on next page

Table 3 – continued from previous page

Country	Document Type
Thailand	
Turkey	
Ukraine	

CHAPTER

TEN

RESEARCH FOR THE PROJECT

10.1 Postal Code

Research about the different postal code format for the different countries to be used in the application. it's required to create regular expression to validate the postal code for each country.

10.1.1 AF - Afghanistan

- Afghanistan postal codes are four digits (NNNN). The first two digits (ranging from 10–43) correspond to the province, while the last two digits correspond either to the city/delivery zone (range 01–50) or to the district/delivery zone (range 51–99).

```
Example:  1001
```

Address Format

```
Mr Ahmad Towheed
Street, House No. 240
Kabul
1001
AFGHANISTAN
```

10.1.2 IN - INDIA

- Indian postal codes are six digits. The first digit represents the region in India. The second digit is the sub-region, while the third digit is the sorting district. The last three digits represent the particular post office within the district.

```
Example:  110001
```

Address Format

```
Mr I.K. Taneja
Flat No.100
Triveni Apartments
Pitam Pura
NEW DELHI 110034
INDIA
```

10.1.3 US - USA

- A United States postal code, it included the five digits of the ZIP Code, followed by a hyphen and four digits that designated a more specific location.

Example: 10001-1234

Address Format

```
Embassy of the United States of America
9000 New Delhi Place
Washington, DC  20521-9000[22]
```

10.1.4 SA - Saudi Arabia

- 5 digits to the right of the locality followed by 4 digits giving additional building information, separated by a dash.eg: The first digit represents the postal region,the second digit represents the sector,the third digit represents the branch,the fourth digit represents the section,the fifth digit represents the block,and the last four digits represent the extended postcode.

Example:- 12233 - 7318

Address Format

```
Mr. Abdullah Nasser
3909 Abdullhamid Alkatib ST, Azahrah Dist.
RIYADH 12987 - 7318
SAUDI ARABIA
```

10.1.5 ZA - South Africa

The postal codes in South African consist of four digits.

Example: 2190

Address Format

```
47 Rockey Street
YEOVILLE
2198
```

10.1.6 UK - United Kingdom

- The UK postcode consists of five to seven alphanumeric characters which was created by Royal Mail. A full postcode designates an area with multiple addresses or a single delivery point.

Example: SW1A 2AA

10.1.7 SG - Singapore

Singapore postal codes are six digits. The first two digits represent the district, while the last four digits represent the street.

Example: 059532

Address Format

```
Ms. Tan Bee Soo
16 Sandilands Road
SINGAPORE 546080
REP. OF SINGAPORE
```

10.1.8 PK - Pakistan

The postal codes in Pakistan consist of 5 digits. The first two digits represent the routeing district, the last three digits represent the post office.

Example: 44000

Address Format

```
Pakistan Museum of Natural History
Garden Avenue
Shakarparian
Islamabad 44000
PAKISTAN
```