Sure, here are some Python regular expressions (regex) for the patterns you mentioned. I’ll also provide some valid and invalid examples for testing.

1. **Email format:**

import re

def test\_email(email):

pattern = r"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$"

if re.match(pattern, email):

return True

return False

# Valid email

print(test\_email("example@example.com")) # Expected output: True

# Invalid email

print(test\_email("example@.com")) # Expected output: False

1. **Mathematical expression:**

def test\_math\_expr(expr):

pattern = r"^[\d+\-\*\/\(\) ]+$"

if re.match(pattern, expr):

return True

return False

# Valid expression

print(test\_math\_expr("(3 + 2) \* 5")) # Expected output: True

# Invalid expression

print(test\_math\_expr("(3 + a) \* 5")) # Expected output: False

1. **URL:**

def test\_url(url):

pattern = r"http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+"

if re.match(pattern, url):

return True

return False

# Valid URL

print(test\_url("https://www.example.com")) # Expected output: True

# Invalid URL

print(test\_url("htp:/example.com")) # Expected output: False

1. **Zip Code (assuming US format):**

def test\_zip(zip\_code):

pattern = r"^\d{5}(?:[-\s]\d{4})?$"

if re.match(pattern, zip\_code):

return True

return False

# Valid zip code

print(test\_zip("12345")) # Expected output: True

# Invalid zip code

print(test\_zip("1234")) # Expected output: False

1. **Phone number (assuming US format):**

def test\_phone(phone):

pattern = r"^(?:\+?1\s?)?((?:[a-z2-9][a-z0-9]{2})|(?:\([a-z2-9][a-z0-9]{2}\)))[-\s]?(\d{3})[-\s]?(\d{4})$"

if re.match(pattern, phone):

return True

return False

# Valid phone number

print(test\_phone("(123) 456-7890")) # Expected output: True

# Invalid phone number

print(test\_phone("123-45-67890")) # Expected output: False

Please replace a-z with ا-ی and A-Z with آ-ی in the regex patterns for Persian language support.

Sure, I’d be happy to explain the regular expressions (regex) used in each case:

1. **Email format:**

pattern = r"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$"

Explanation:

* ^ : Start of the line.
* [a-zA-Z0-9\_.+-]+ : One or more (+) of the characters inside the square brackets ([]). These characters can be lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), underscore (\_), dot (.), plus (+), or hyphen (-).
* @ : Matches the @ symbol.
* [a-zA-Z0-9-]+ : One or more (+) of the characters inside the square brackets ([]). These characters can be lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), or hyphen (-).
* \. : Matches the dot symbol (.).
* [a-zA-Z0-9-.]+ : One or more (+) of the characters inside the square brackets ([]). These characters can be lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), dot (.), or hyphen (-).
* $ : End of the line.

1. **Mathematical expression:**

pattern = r"^[\d+\-\*\/\(\) ]+$"

Explanation:

* ^ : Start of the line.
* [\d+\-\*\/\(\) ]+ : One or more (+) of the characters inside the square brackets ([]). These characters can be digits (\d), plus (+), hyphen (-), asterisk (\*), slash (/), parentheses (()), or space.
* $ : End of the line.

1. **URL:**

pattern = r"http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+"

Explanation: This pattern is a bit more complex, but here’s a breakdown:

* http[s]?:// : Matches “http://” or “https://”.
* (?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+ : One or more (+) of any of these groups. The groups are separated by vertical bars (|) which act like an OR operator. The groups include lowercase letters (a-z), uppercase letters (A-Z), numbers (0-9), special symbols ($-\_@.&+), special symbols enclosed in parentheses (!\*\(\)), and hexadecimal values (%[0-9a-fA-F][0-9a-fA-F]).

1. **Zip Code:**

pattern = r"^\d{5}(?:[-\s]\d{4})?$"

Explanation:

* ^ : Start of the line.
* \d{5} : Exactly five digits.
* (?:[-\s]\d{4})? : An optional group (?) that includes a hyphen (-) or space (\s) followed by exactly four digits (\d{4}).
* $ : End of the line.

1. **Phone number:**

pattern = r"^(?:\+?1\s?)?((?:[a-z2-9][a-z0-9]{2})|(?:\([a-z2-9][a-z0-9]{2}\)))[-\s]?(\d{3})[-\s]?(\d{4})$"

Explanation: This pattern is also complex, but here’s a breakdown:

* ^ : Start of the line.
* (?:\+?1\s?)? : An optional group (?) that includes an optional plus sign (+?), followed by 1, followed by an optional space (\s?). The rest of the pattern matches a phone number in various formats, such as “1234567890”, “123 456 7890”, “(123)4567890”, “(123) 456 7890”, etc.

I hope this helps! Let me know if you have any other questions. 😊

Sure, here’s a modified version of the phone number regex that also checks for the country code:

def test\_phone(phone):

pattern = r"^\+\d{1,3}\s?((?:[a-z2-9][a-z0-9]{2})|(?:\([a-z2-9][a-z0-9]{2}\)))[-\s]?(\d{3})[-\s]?(\d{4})$"

if re.match(pattern, phone):

return True

return False

# Valid phone number

print(test\_phone("+1 (123) 456-7890")) # Expected output: True

print(test\_phone("+98 123 456 7890")) # Expected output: True

# Invalid phone number

print(test\_phone("123-45-67890")) # Expected output: False

In this pattern, ^\+\d{1,3}\s? checks for the country code at the beginning of the string. It matches a plus sign (+) followed by one to three digits (\d{1,3}), and an optional space (\s?). The rest of the pattern is the same as before. Please replace a-z with ا-ی and A-Z with آ-ی in the regex patterns for Persian language support. Let me know if you have any other questions! 😊

Sure, here’s a modified version of the zip code regex that matches the Iranian zip code format:

def test\_zip(zip\_code):

pattern = r"^\d{5}[-\s]?\d{5}$"

if re.match(pattern, zip\_code):

return True

return False

# Valid zip codes

print(test\_zip("1616856761")) # Expected output: True

print(test\_zip("16168 56761")) # Expected output: True

print(test\_zip("16168-56761")) # Expected output: True

# Invalid zip code

print(test\_zip("1234")) # Expected output: False

Explanation of the pattern r"^\d{5}[-\s]?\d{5}$":

* ^ : Start of the line.
* \d{5} : Exactly five digits, representing the departure code.
* [-\s]? : An optional group (?) that includes a hyphen (-) or space (\s).
* \d{5} : Exactly five digits, representing the distribution code.
* $ : End of the line.

This pattern will match a 10-digit number with an optional space or hyphen between the 5th and 6th digits. Let me know if you have any other questions! 😊