# Assignment 09 Solution

Submitted by: Yameen Ali

Go to Github repository

---

# Question 1:

## What metacharacters are in regular expressions and provide examples of commonly used metacharacters. Discuss their significance in pattern matching?

---

metacharacters:

```
In regular expressions, metacharacters are special characters that have a specific meaning or
function rather than representing themselves literally. They are crucial for pattern matching
because they allow you to define complex search patterns. Here are some commonly used
metacharacters and their significance:
```

In [13]:
```
import re
```

### . (dot):

Matches any single character except newline. For example, the pattern b.t would match "bat", "bit", "bot", etc., but not "bat" in "bathe".

In [15]:
```
pattern_dot = re.compile(r'b.t')
print(pattern_dot.findall('bat,bit,ball'))
```

```
['bat', 'bit']
```

### ^ (caret):

Matches the start of a line. When used outside of a character class, it anchors the pattern to the beginning of a line. For example, the pattern ^cat would match "cat" at the beginning of a line.

In [14]:
```
pattern_caret = re.compile(r'^cat')
print(pattern_caret.findall('cat is cute\nconcatenate'))
```

```
['cat']
```

### $$ (dollar):

Matches the end of a line. It anchors the pattern to the end of a line. For example, the pattern dog$ would match "dog" only at the end of a line.

In [27]:
```
pattern_dollar = re.compile(r'yameen$')
print(pattern_dollar.findall('yameen is a\npython developer yameen'))
```

```
['yameen']
```

### [ ] (square brackets):

Defines a character class. It matches any single character within the brackets. For example, [aeiou] matches any vowel.

In [28]:
```
pattern_char_class = re.compile(r'[aeiou]')
print(pattern_char_class.findall('apple, banana'))
```

```
['a', 'e', 'a', 'a', 'a']
```

### | (vertical bar):

Represents alternation. It allows you to specify multiple alternatives. For example, cat|dog matches either "cat" or "dog".

In [29]:
```
pattern_alternation = re.compile(r'cat|dog')
print(pattern_alternation.findall('I have a cat and a dog'))
```

```
['cat', 'dog']
```

### * (asterisk):

Matches zero or more occurrences of the preceding element. For example, ab* matches "a", "ab", "abb", "abbb", etc.

In [31]:
```
pattern_asterisk = re.compile(r'ab*')
print(pattern_asterisk.findall('a, ab, abb, abbb,abbbbbbbbbbbb'))
```

```
['a', 'ab', 'abb', 'abbb', 'abbbbbbbbbbbb']
```

### + (plus):

Matches one or more occurrences of the preceding element. For example, ab+ matches "ab", "abb", "abbb", etc., but not "a".

In [33]:
```
pattern_plus = re.compile(r'ab+')
print(pattern_plus.findall('a, ab, abb, abbb'))
```

```
['ab', 'abb', 'abbb']
```

### ? (question mark):

Matches zero or one occurrence of the preceding element. For example, ab? matches "a" or "ab".

In [34]:
```
pattern_question = re.compile(r'ab?')
print(pattern_question.findall('a, ab, abb, abbb'))
```

```
['a', 'ab', 'ab', 'ab']
```

### { } (curly braces):

Specifies the exact number of occurrences or a range of occurrences of the preceding element. For example, a{2} matches "aa", and b{2,4} matches "bb", "bbb", or "bbbb".

In [35]:
```
pattern_curly_braces = re.compile(r'a{2}')
print(pattern_curly_braces.findall('a, aa, aaa'))
```

```
['aa', 'aa']
```

### \ (backslash):

Escapes a metacharacter, allowing you to match it literally. For example, . matches a period character instead of acting as a metacharacter.

In [36]:
```
pattern_escape = re.compile(r'\.')
print(pattern_escape.findall('a.b, c.d'))
```

```
['.', '.']
```

---

# Question2: (Email Validation for User Registration)

Many websites require users to register using their email addresses. Develop a Python script that validates email addresses using regular expressions. Consider various email formats and ensure that the script can accurately distinguish between valid and invalid email addresses.

In [42]:
```
import re

def validate_email(email):
    # Regular expression for validating email addresses
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

    # Compile the regular expression pattern
    regex = re.compile(pattern)

    # Check if the email matches the pattern
    if regex.match(email):
        return True
    else:
        return False

# Test email addresses
emails = [
    "yameenshah012@gmail.com",
    "user@example.com",
    "user123@gmail.com",
    "user.name@domain.co.uk",
    "user123@sub.domain.com",
    "123user@domain.com",
    "user123@123domain.com",
    "user@localhost",
    "user@invalid",
```

```
    "user@.com",
    "user@domain",
    "user@example",
    "user123@",
    "invalidemail@.com",
    "@domain.com",
    "user@123.456",
    "user123@domain..com"

]

# Validate and print the result for each email
for email in emails:
    print(f"{email}: {'Valid' if validate_email(email) else 'Invalid'}")
```

```
yameenshah012@gmail.com: Valid
user@example.com: Valid
user123@gmail.com: Valid
user.name@domain.co.uk: Valid
user123@sub.domain.com: Valid
123user@domain.com: Valid
user123@123domain.com: Valid
user@localhost: Invalid
user@invalid: Invalid
user@.com: Invalid
user@domain: Invalid
user@example: Invalid
user123@: Invalid
invalidemail@.com: Invalid
@domain.com: Invalid
user@123.456: Invalid
user123@domain..com: Valid
```

---

# Question3: (User Log Analysis for Error Detection)

Analyze log files from a web server or application and extract relevant information using regular expressions. Develop a Python script that reads log files, identifies error messages based on predefined patterns, and generates a summary report with details of errors encountered.

In [43]:

```
import re

def analyze_logs(log_data):
    error_pattern = r'\[(ERROR)\] (.+)'
    error_count = 0
    error_details = []

    for log_entry in log_data:
        match = re.search(error_pattern, log_entry)
        if match:
            error_count += 1
            error_details.append((match.group(1), match.group(2)))

    return error_count, error_details

# Sample log data
log_data = [
    "2024-04-18 08:30:45 [INFO] Application started successfully.",
    "2024-04-18 08:32:15 [ERROR] Database connection failed: Connection timed out.",
    "2024-04-18 08:35:22 [ERROR] Failed to process request: Invalid input data.",
    "2024-04-18 08:40:11 [WARNING] Disk space running low: 90% usage detected.",
    "2024-04-18 08:45:03 [ERROR] Server crashed: Segmentation fault.",
    "2024-04-18 08:48:55 [INFO] New user registered: username123.",
    "2024-04-18 08:50:27 [ERROR] Authentication failed: Invalid credentials.",
    "2024-04-18 08:55:09 [ERROR] Internal server error: Unable to connect to external service.",
    "2024-04-18 09:00:04 [INFO] Application stopped."
]

# Analyze logs
error_count, error_details = analyze_logs(log_data)

# Generate summary report
print("Summary Report:")
print(f"Total Errors Encountered: {error_count}")
print("Error Details:")
for severity, message in error_details:
    print(f"- [{severity}]: {message}")
```

```
Summary Report:
Total Errors Encountered: 5
Error Details:
- [ERROR]: Database connection failed: Connection timed out.
- [ERROR]: Failed to process request: Invalid input data.
- [ERROR]: Server crashed: Segmentation fault.
- [ERROR]: Authentication failed: Invalid credentials.
- [ERROR]: Internal server error: Unable to connect to external service.
```

---

# Question4: (Web Scraping and Information Retrieval)

Implement a web scraping tool that retrieves information from web pages based on specified patterns using regular expressions. Develop a Python script that accesses web pages, extracts desired content (e.g., product prices, article titles), and stores the extracted data for further analysis.

In [22]:
```python
import requests
import re
import json

# Function to scrape book information using regular expressions
def scrape_books(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
    }

    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        page_content = response.text

        # Regular expression to find book titles
        title_pattern = re.compile(r'<h3><a href="[^"]+" title="([^"]+)">')
        titles = title_pattern.findall(page_content)

        # Regular expression to find book prices
        price_pattern = re.compile(r'<p class="price_color">([^<]+)</p>')
        prices = price_pattern.findall(page_content)

        # Combine titles and prices
        books = [{"title": title, "price": price} for title, price in zip(titles, prices)]

        # Store the scraped data in a JSON file
        with open('scraped_books.json', 'w', encoding='utf-8') as f:
            json.dump(books, f, ensure_ascii=False, indent=4)

        return books
    else:
        print("Failed to retrieve the webpage.")
        return []

# URL of the webpage to scrape
url = 'https://books.toscrape.com/catalogue/category/books_1/index.html'
books = scrape_books(url)

# Display the scraped book information
for book in books:
    print(f"Book Title: {book['title']}, Price: {book['price']}")
```

```
Book Title: A Light in the Attic, Price: Â£51.77
Book Title: Tipping the Velvet, Price: Â£53.74
Book Title: Soumission, Price: Â£50.10
Book Title: Sharp Objects, Price: Â£47.82
Book Title: Sapiens: A Brief History of Humankind, Price: Â£54.23
Book Title: The Requiem Red, Price: Â£22.65
Book Title: The Dirty Little Secrets of Getting Your Dream Job, Price: Â£33.34
Book Title: The Coming Woman: A Novel Based on the Life of the Infamous Feminist, Victoria Woodhull, Price: Â£1
7.93
Book Title: The Boys in the Boat: Nine Americans and Their Epic Quest for Gold at the 1936 Berlin Olympics, Pri
ce: Â£22.60
Book Title: The Black Maria, Price: Â£52.15
Book Title: Starving Hearts (Triangular Trade Trilogy, #1), Price: Â£13.99
Book Title: Shakespeare&#39;s Sonnets, Price: Â£20.66
Book Title: Set Me Free, Price: Â£17.46
Book Title: Scott Pilgrim&#39;s Precious Little Life (Scott Pilgrim #1), Price: Â£52.29
Book Title: Rip it Up and Start Again, Price: Â£35.02
Book Title: Our Band Could Be Your Life: Scenes from the American Indie Underground, 1981-1991, Price: Â£57.25
Book Title: Olio, Price: Â£23.88
Book Title: Mesaerion: The Best Science Fiction Stories 1800-1849, Price: Â£37.59
Book Title: Libertarianism for Beginners, Price: Â£51.33
Book Title: It&#39;s Only the Himalayas, Price: Â£45.17
```

## By Using BeautifulSoup Library

## By Using BeautifulSoup Library

In [8]:
```
import requests
from bs4 import BeautifulSoup

# Function to scrape book information
def scrape_books(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/

    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        soup = BeautifulSoup(response.content, "html.parser")

        # Find all book blocks
        book_blocks = soup.find_all("article", class_="product_pod")

        books = []
        for book in book_blocks:
            title = book.h3.a["title"]
            price = book.find("p", class_="price_color").text
            books.append({"title": title, "price": price})

        return books
    else:
        print("Failed to retrieve the webpage.")
        return []

# URL of the webpage to scrape
url = 'https://books.toscrape.com/catalogue/category/books_1/index.html'
books = scrape_books(url)

# Display the scraped book information
for book in books:
    print(f"Book Title: {book['title']}, Price: {book['price']}")
```

```
Book Title: A Light in the Attic, Price: £51.77
Book Title: Tipping the Velvet, Price: £53.74
Book Title: Soumission, Price: £50.10
Book Title: Sharp Objects, Price: £47.82
Book Title: Sapiens: A Brief History of Humankind, Price: £54.23
Book Title: The Requiem Red, Price: £22.65
Book Title: The Dirty Little Secrets of Getting Your Dream Job, Price: £33.34
Book Title: The Coming Woman: A Novel Based on the Life of the Infamous Feminist, Victoria Woodhull, Price: £17
.93
Book Title: The Boys in the Boat: Nine Americans and Their Epic Quest for Gold at the 1936 Berlin Olympics, Pri
ce: £22.60
Book Title: The Black Maria, Price: £52.15
Book Title: Starving Hearts (Triangular Trade Trilogy, #1), Price: £13.99
Book Title: Shakespeare's Sonnets, Price: £20.66
Book Title: Set Me Free, Price: £17.46
Book Title: Scott Pilgrim's Precious Little Life (Scott Pilgrim #1), Price: £52.29
Book Title: Rip it Up and Start Again, Price: £35.02
Book Title: Our Band Could Be Your Life: Scenes from the American Indie Underground, 1981-1991, Price: £57.25
Book Title: Olio, Price: £23.88
Book Title: Mesaerion: The Best Science Fiction Stories 1800-1849, Price: £37.59
Book Title: Libertarianism for Beginners, Price: £51.33
Book Title: It's Only the Himalayas, Price: £45.17
```

In [ ]: