Regular Expressions (RegEx)

Def - Sequence of characters that define a search pattern in a given string/file.

Applications:

- Search for particular patterns in string
- File Processing
- Extract data from log files by matching certain patterns.

In order to work with regular expressions, we need to import module RE.

This RE module provides a set of in-built functions, that enables programmer to search for matching patterns/string.

- compile() – Compile a regular expression pattern as a string into regex pattern object.
- findall() – returns list containing all matches
- split() - returns a list of strings where the matching splits have occurred.

Example:

```
import re
sentence1 = "Python is a programming language"
sentence2 = "Hello 22 world 44 Welcome 66"
pattern= re.compile("\d+")
result1 = pattern.findall(sentence1)
result2 = pattern.findall(sentence2)
print(result1)
print(result2)
print(re.findall('\d+',sentence2))
print(re.split('\d{3}',sentence2))
print(re.findall('\d{2}',sentence2))
```

- match() – checks for the matching pattern at start of the string.
- search() – checks for the matching pattern at anywhere in the string.

Example:

```
print(re.match("Python",sentence1))
print(re.search("program",sentence1).group())
```

```
print(re.match(r"\w{6}",sentence1))
print(re.search(r"\w{2}", sentence1).group())
```

- sub() – substitute/replace pattern in a string

Example:

```
print(re.sub("\s+","_", sentence1))
print(re.sub("\s","", sentence1))
```

Every character in a RegEx is either a metacharacter or a regular character.

Meta character: A special character which has its own meaning.

Regular character: A character which matches itself.

Most Common meta characters:

1. "^" – using this we can check whether a string starts with particular character/word.
   Ex: print(re.search(r"^Python",sentence1).group())
2. "$" - using this we can check whether a string ends with particular character/word.
   Ex: print(re.search(r"language$",sentence1).group())
3. "*" – using this we can find if there are 0 or more repetitions
4. "+" - using this we can find if there are 1 or more repetitions
5. "?" - using this we can find if there are 0 or 1 repetitions
   Example:
   ```
   import re
   sentence2 = "Hel2546lo 22 wo9rld 44 Wel456come 66"
   print(re.findall(r"\d\d*",sentence2))
   print(re.findall(r"\d\d+", sentence2))
   print(re.findall(r"\d\d\d\d\d?", sentence2))
   ```
6. "." – used to find exact end of sentence
7. "[]" – checks for the particular set of characters.
   ```
   sentence3 = "Hello Customer. Welcome to ABC hotel. Have a nice stay."
   print(re.findall(r"\.", sentence3))
   print(re.findall(r"[ace]", sentence3))
   ```

Character Classes

Sets/Ranges of characters enclosed by square brackets.

Ex : [abc], [a-z]

[abc] - Match the letter a/b/c

[^abc] – Match the letter other than a,b,c

[agt][eo] – Match the letter a/g/t followed by e/o

[0-9] – Match any digit between 0 to 9

[a-z] – Match any lowercase letters

[A-Z] – match any uppercase letters

[a-zA-Z0-9] – Match any alphanumeric character

[a-q3-9] – Match the letter between a-q & digit between 3 to 9.

Special Sequences

- Defines the basic predefined character class patterns.
- Represents a special meaning
- Consists of a special character prefixed with backslash

Ex: \A, \s

\A – Match pattern at start of string

\Z – Match pattern at end of string

\d – Match any digit

\D – Match any non digit

\s – Match any space

\S – Match any non space

\w – Match any alphanumeric character

\W – Match any alphanumeric character

```
line1 = "Arjun is a Infosys employee bearing employee id of 35489"
line2 = "His address is 11/23/345, Flat no 303, 65 Hills Singapore"
print(re.findall(r'\A([A-Z].*?)\s',line1))
print(re.findall(r'\A(A...n)\S',line1))
print(re.findall(r'\d+\Z',line1))
print(re.findall(r'\D+\Z',line2))
print(re.findall(r"\w{7}",line1))
print(re.findall(r"[0-9]",line2))
print(re.findall(r"[g-l1-4]",line2))
```

Exception Handling

- Exceptions are raised when program faces an error
- If error is not handled properly then the python interpreter will stop process execution.
- To ensure the flow of program to run without interruptions, there is need of handling these exceptions.

Syntax:

```
try:
  # do something
  pass
except ValueError:
  # handle ValueError exception
  pass
except (TypeError, ZeroDivisionError):
  # handle multiple exceptions
  # TypeError and ZeroDivisionError
  pass
except:
  # handle all other exceptions
  pass
finally:
```

```
    # execute always
```

Example:

```
class ValueSmallError(Exception):
    pass
try:
    n1 = int(input("Enter the number1 : "))
    n2 = int(input("Enter the number2 : "))
    if n1/n2 > 1:
        print(f"{n1} is greater than {n2}")
    else:
        raise ValueSmallError
except ValueError:
    print("Input should be of integer type")
except ZeroDivisionError:
    print("Division cannot be possible with zero")
except ValueSmallError:
    print("n1 value must be greater than n2")
finally:
    print("Exception Handling is implemented")
```