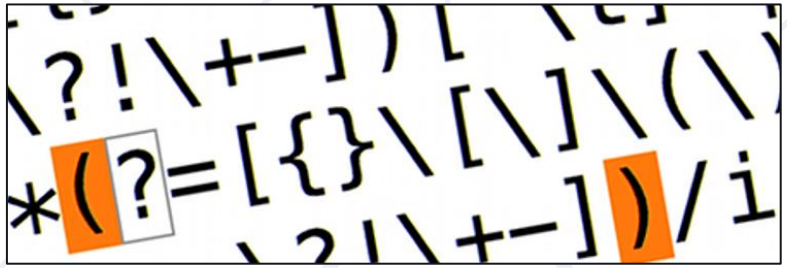
Regular Expressions



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#fund-python

Table of Contents

1. What is **RegEx**?
2. **Syntax**
3. RegEx in Python
4. RegEx **Method**

What is RegEx?

What is RegEx?

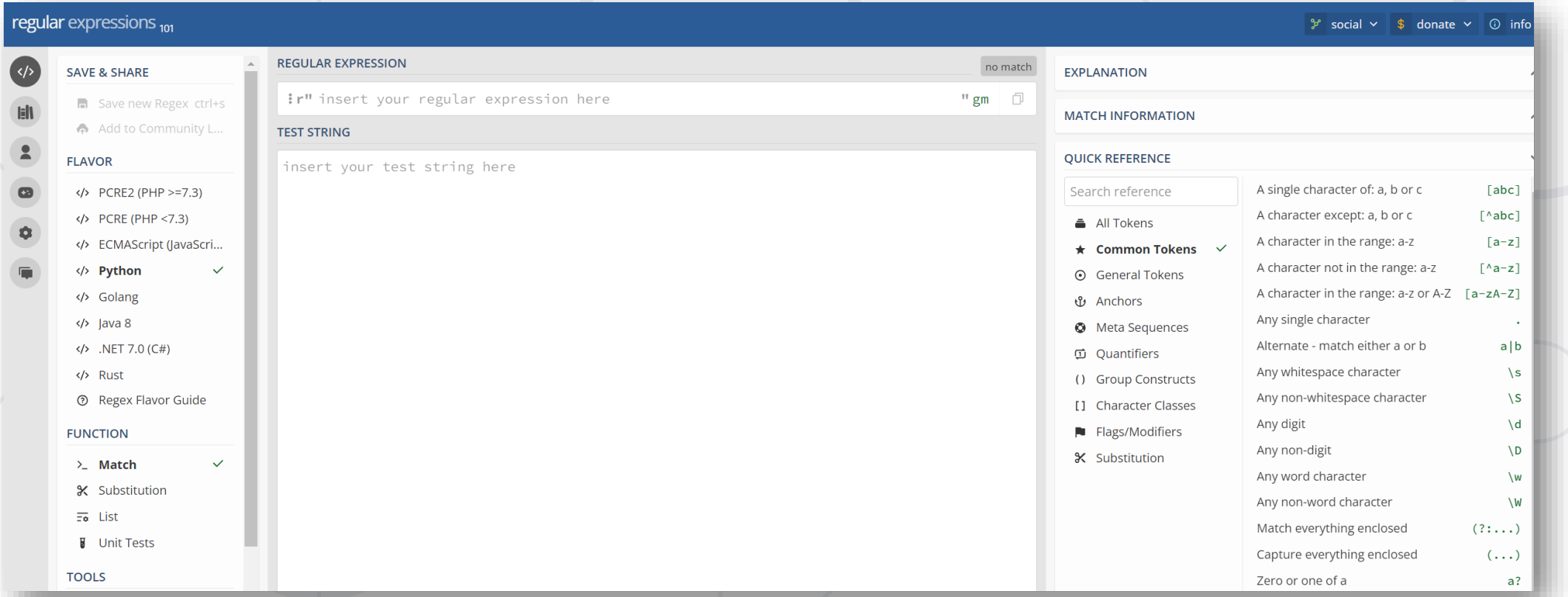
- A regular expression or regex is a sequence of characters that define a **search pattern**
- Usually, such patterns are used by string **searching** algorithms for "find" or "find and replace" **operations** on strings
- Regular expressions are used in search engines, search and replace dialogs of word processors and text editors etc

Example

- For example, if you want to **validate** if an **email** is valid, it should follow these rules:
  - have only **alphanumeric** characters
  - include **"@"**
  - end with **.com/.bg/.net**
- To validate emails will be very difficult without regular expression

Regex101

- To **test** Regular Expressions, use ***regex101***



Syntax

Special Sequences

Notation	Meaning
\d	Returns a match where the string contains **digits** (numbers from 0-9)
\D	Returns a match where the string **DOES NOT** contain digits
\b	Returns a match where the specified characters are at the **beginning** or at the **end** of a word
\s	Returns a match where the string contains a **white space** character
\S	Returns a match where the string **DOES NOT** contain a white space character
\w	Returns a match where the string contains any **word** characters (characters from a to Z, digits from 0-9, and the underscore _ character)
\W	Returns a match where the string **DOES NOT** contain any word characters

Metacharacters

Software University

Notation	Meaning
\	Signals a special sequence (can also be used to escape special characters)
.	Any character (except newline character)
+	One or more occurrences
*	Zero or more occurrences
|	Either or
()	Capture and group
{}	Exactly the specified number of occurrences
^	Starts with
$	Ends with

Sets

Examples	Meaning
[arn]	Returns a match where **one** of the specified characters (a, r, or n) are present
[a-n]	Returns a match for any **lower-case** character, alphabetically between a and n
[^arn]	Returns a match for any character **EXCEPT** a, r, and n
[0123]	Returns a match where **any** of the specified digits (0, 1, 2, or 3) are present
[0-9]	Returns a match for any digit **between** 0 and 9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character **alphabetically** between a and z, lower case OR upper case

Problem: Match All Words

- Write a regular expression in **regex101** that extracts all word char sequences from given text

```
_ (Underscores) are also word characters!
```

```
_|Underscores|are|also|word|characters
```

Problem: Match Dates

- Write a regular expression that extracts **dates** from text
  - Valid date format: **dd-MMM-yyyy**
  - Examples: **12-Jun-1999**, **3-Nov-1999**

```
I am born on 30-Dec-1994.
My father is born on the 9-Jul-1955.
01-July-2000 is not a valid date.
```

Problem: Email Validation

- Write a regular expression that performs simple **email validation**
  - An email consists of **username @ domain name**
  - **Usernames** are **alphanumeric**
  - **Domain names** consist of **two strings**, separated by a **period**
  - **Domain names** may contain only **English letters**

Valid:

```
valid123@email.bg
```

Invalid:

```
invalid*name@emai1.bg
```

RegEx in Python

RegEx Module

- Python has a built-in package called re
- It can be used to work with Regular Expressions
- Import the **re** module

```
import re
```

- Use it to search in text

```
import re
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)
```

RegEx Methods

findall() Method

- The `findall()` function returns a list containing all matches

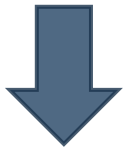
```
import re
str = "The rain in Spain"
x = re.findall("ai", str)
print(x) # ["ai", "ai"]
```

- The list contains the matches in the **order** they are **found**
- If **no matches** are found, an **empty list** is returned

Problem: Match Full Name

- You are given a list of names
  - Match all full names

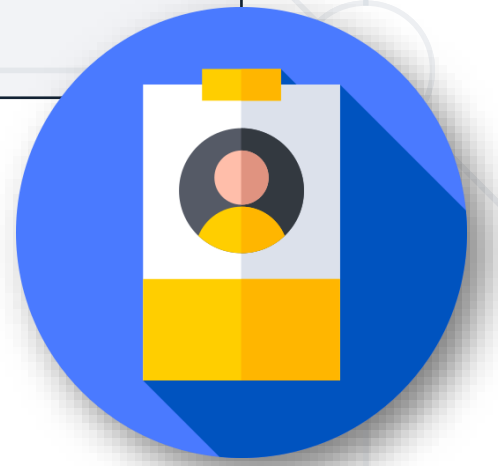
```
Bethany Taylor, Oliver miller, sophia Johnson, SARah
Wilson, John Smith, Sam      Smith
```



```
Bethany Taylor John Smith
```

Solution: Match Full Name

```
import re
names = input()
regex = "\\b[A-Z][a-z]+ [A-Z][a-z]+\\b"
matches = re.findall(regex, names)
print(" ".join(matches))
```



Problem: Match Phone Number

- You are given a list of phone numbers
  - Match all valid phone numbers

```
+359 2 222 2222,359-2-222-2222, +359/2/222/2222,
+359-2 222 2222 +359 2-222-2222, +359-2-222-222,
+359-2-222-22222 +359-2-222-2222
```

```
+359 2 222 2222, +359-2-222-2222
```

Solution: Match Phone Number

```
import re
pattern = "(\+359-2-[0-9]{3}-[0-9]{4}\\b|\+359 2 [0-9]{3}
[0-9]{4})\\b"
text = input()
matches = re.findall(pattern, text)
print(", ".join(matches))
```

Problem: Match Dates

- Write a program, which matches a date in the format `"dd{separator}MMM{separator}yyyy"`
  - Use **capturing groups** in your regular expression

```
13/Jul/1928, 10-Nov-1934, , 01/Jan-
1951,f 25.Dec.1937 23/09/1973,
1/Feb/2016
```

```
Day: 13, Month: Jul, Year: 1928
Day: 10, Month: Nov, Year: 1934
Day: 25, Month: Dec, Year: 1937
```

Solution: Match Dates

```
import re
pattern = "\\b(?P<day>\\d{2})([-.\\/])(?P<month>[A-Z]
[a-z]{2})\\2(?P<year>\\d{4})\\b"
text = input()
matches = re.finditer(pattern, text)
for match in matches:
    print(f"Day: {match.group('day')}, Month:
{match.group('month')}, Year: {match.group('year')}")
```

search() Method

- The `search()` function searches the string for a match and returns a **Match object** if there is a match
- If there is more than one match, only the **first occurrence** of the match will be returned

```
import re
str = "The rain in Spain"
x = re.search("\s", str)
print("The first white-space character is located in position:", x.start())
```

- If no matches are found, the value **None** is returned

split() Method

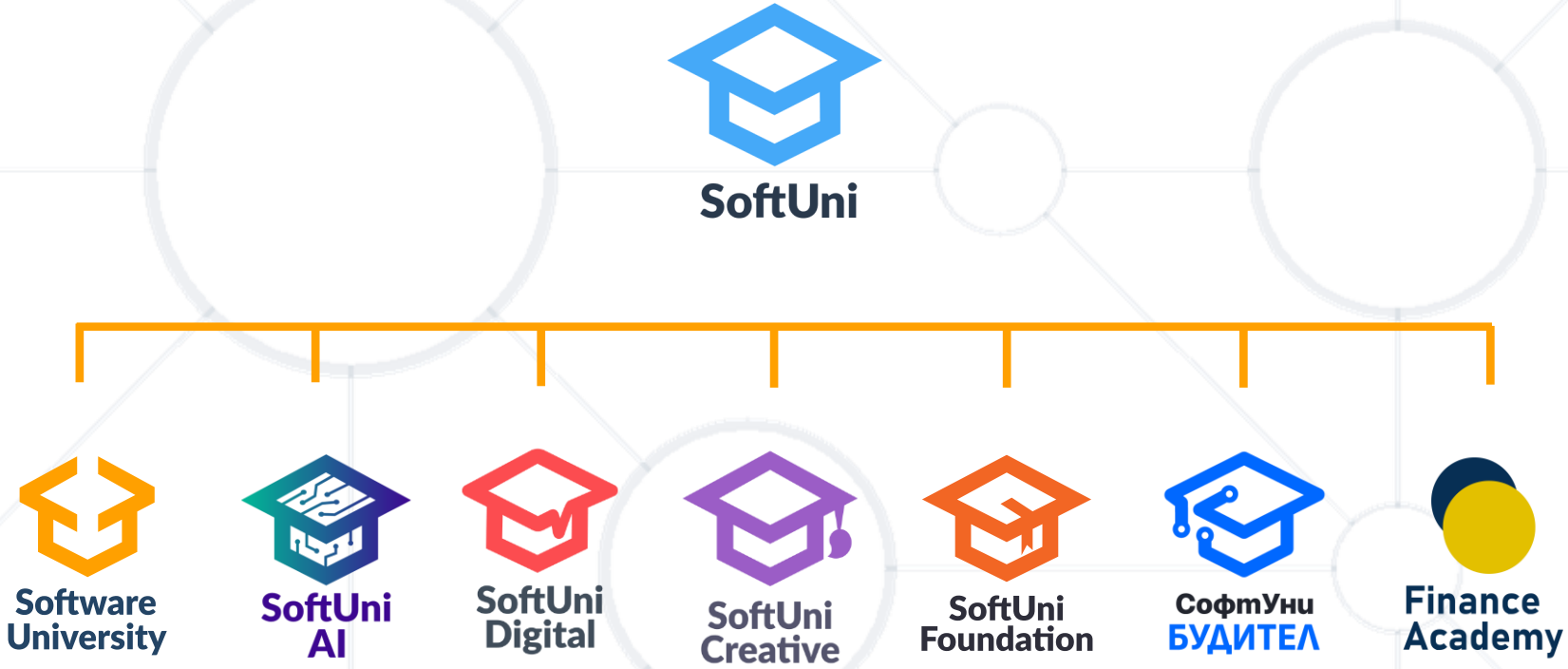
- The `split()` function returns a list where the string has been split at each match

```
import re
str = "The rain in Spain"
x = re.split("\s", str)
print(x)
# ['The', 'rain', 'in', 'Spain']
```

Summary

- A regular expression or regex is a sequence of characters that define a search pattern
- Python has a built-in package called re
- It can be used to work with Regular Expressions

Questions?



© SoftUni – https://about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

SoftUni Diamond Partners

Software University

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

createX

DRAFT KINGS

THE CROWN IS YOURS

INDEAVR

Serving the high achievers

encorp.io

VIVACOM

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg