|  |  |  |
| --- | --- | --- |
| **College of Engineering**  Computer Science & Eng. Dept.  **Course:** CMP 321L Programminglanguages Lab | A picture containing logo  Description automatically generated | **Course Professor:** Dr. Michel Pasquier  **Lab Instructor:** Praveena Kolli  **Office:** EB2-126  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Summer 2022 |

**Lab 7 – Python Regular Expressions**

**Objectives:**

* Understand the syntax and usage of regular expressions
* Code Python scripts that make use of regular expressions

**Due date: End of the lab. (**Only one team member needs to submit.)

**Rules:**

(1) Usage: **You should explore and make good use of the Python features you learned in class.** (2) Scope: **You should only use those features that have been explained in detail in class.**

(3) Style: Follow standard Python programming style and conventions.

(4) Logic: Add appropriate comments to your code to explain your solution.

(Code / answers that do not follow the above specifications will be penalized.)

***Warning:* You need to come to the lab properly prepared i.e.**

(1) Make sure you have studied and understood the class material.

(2) Read the lab doc, think about the problems, and prepare questions as needed.

If you do not, completing the lab in 2.45 hours may become too much of a challenge!

**Useful resources:**

<http://regex101.com/#python>

<https://docs.python.org/3/library/>

<https://docs.python.org/3/library/re.html>

**Lab By:**

Rohan Mitra 85023

Keshav Ramesh 88595

**Github link for code**: <https://github.com/ro1406/CMP321Lab>

**Note:** Formal grammars for both Exercises 1 and 2 are provided in the appendix.

**Exercise 1: Validating phone numbers format using regex [4 marks]**

Write a Python script that uses a *regular expression* to check if a string represents a valid *phone number,* or not, as per the following specifications (grammar rules). Examples of valid phone numbers include: "(971) 50-5672722", "971-55-6713432", "971-6-5150000, and "056-8887272".

(1) A *phone number* is either of:

– A *country code* in parentheses, followed by a space character, and a *full number;*

– A *country code,* followed by a dash character ('–'), and *full number;*

– A zero ('0'), followed by *full number.*

(2) A *country code* consists of three *digits.*

(3) A *full number* is a *mobile code* or an *area code,* followed by a dash ('–'), and a *number.*

(4) A *mobile code* is either 50 or 55 or 56.

(5) An *area code* is any of 2, 3, 4, 6, 7, or 9.

(6) A *number* consists of seven *digits.*

(7) A *digit* is any from 0 to 9.

**Code:**

import re

def checkPhoneNo(num):

if re.fullmatch(r"((\(\d{3}\) )|(\d{3}-)|0)(50|55|56|2-4|6|7|9)-\d{7}", num):

print(num, "is a valid phone number")

else:

print(num, "is an invalid phone number")

lst = ["(971) 50-5672722", "971-55-6713432", "971-6-5150000" , "056-8887272" ]

for item in lst:

checkPhoneNo(item)

**Screenshot:**

Text

Description automatically generated

**Exercise 2: Validating assignment expressions using regex [6 marks]**

Write a Python script that uses a *regular expression* to check if a given string represents a valid *assignment expression,* or not, as defined by the following specifications (grammar rules). Examples are included hereafter.

(1) An *assignment* expression is:

– An *identifier,* followed by the equal sign ('='), followed by either of an *identifier or a number*

– optionally followed by one or more of : an *operator* followed by either an *identifier* or a *number.*

(2) An *identifier* is a *letter,* optionally followed by one or more of: a *letter* or a *digit.*

(3) A *number* may optionally start with either the plus sign ('+') or minus sign ('–'). It consists of one or more *digits,* optionally followed by: a decimal point ('.') and one or more *digits.*

(4) An operator is either of the symbols for addition ('+'), subtraction ('–'), multiplication ('\*'), division ('/'), or modulus ('%').

(5) A *letter* is any character from the small or uppercase alphabets: 'a-z' or 'A-Z'.

(6) A *digit* is any from 0 to 9.

|  |  |  |  |
| --- | --- | --- | --- |
| *Examples of VALID assignment expressions* |  | *Examples of INVALID assignment expressions* | *Justification / comment* |
| x = y |  | x = y \* | there is no number after the operator |
| y = y \* count2 |  | y = y z | an identifier cannot be followed by another identifier |
| p = 3.1415 \* d |  | 3 = p \* d | the equal sign must be followed by either an identifier or a number |
| z = a + b – 1 / c |  | z = + b – 1 /c | an assignment expression should start with an identifier |

**Code:**

import re

def checkExpression(expression):

if re.fullmatch(r "([a-zA-Z]+[0-9]\*[a-zA-Z]\* = [+-]?\w+(\.[0-9])\*( \*[+\-\*/%] [+-]?\w+(\.[0-9])\*)\*)",expression):

return True

return False

lst = ["x = y", "y = y \* z", "z = a + b - 1 / c", "p = 3.1415 \* d", "x = y \*","y = y z","z = + b – 1 /c","3 = p \* d",'x = -3 + y','x = y / -3'] #test cases for checking expression

for item in lst:

print(item,':',checkExpression(item))

**Screenshot:**

Text, arrow

Description automatically generated

**Appendix:**

**Exercise 1 BNF**

<phone-number> -> ( <country-code> ) " " <number> |

< country-code > - <number> | 0 <number>

<country-code> ->  <digit> <digit> <digit>

<number>  -> <mobile-code> - <digits> | <area-code> - <digits>

<mobile-code> -> 50 | 55 | 56

<area code> -> 2 | 3 | 4 | 6 | 7 | 9

<digits> -> <digit> <digit> <digit> <digit> <digit> <digit> <digit>

<digit>   -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

**Exercise 2 EBNF**

<assignment-statement> -> <identifier> = (<identifier> | <numerical-literal>)

{<operator> (<identifier> | <numerical-literal>)}

<identifier> -> <letter> {<letter> | <digit>}

<numerical-literal> -> [+|-] <digit> {<digit>} [. <digit> {<digit>}]

<digit> -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter> -> a | b | c | d | e | f | g | h | I | j | k | l | m | n | o | p | q

| r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G

| H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W

| X | Y | Z

<operator> -> + | - | \* | / | %