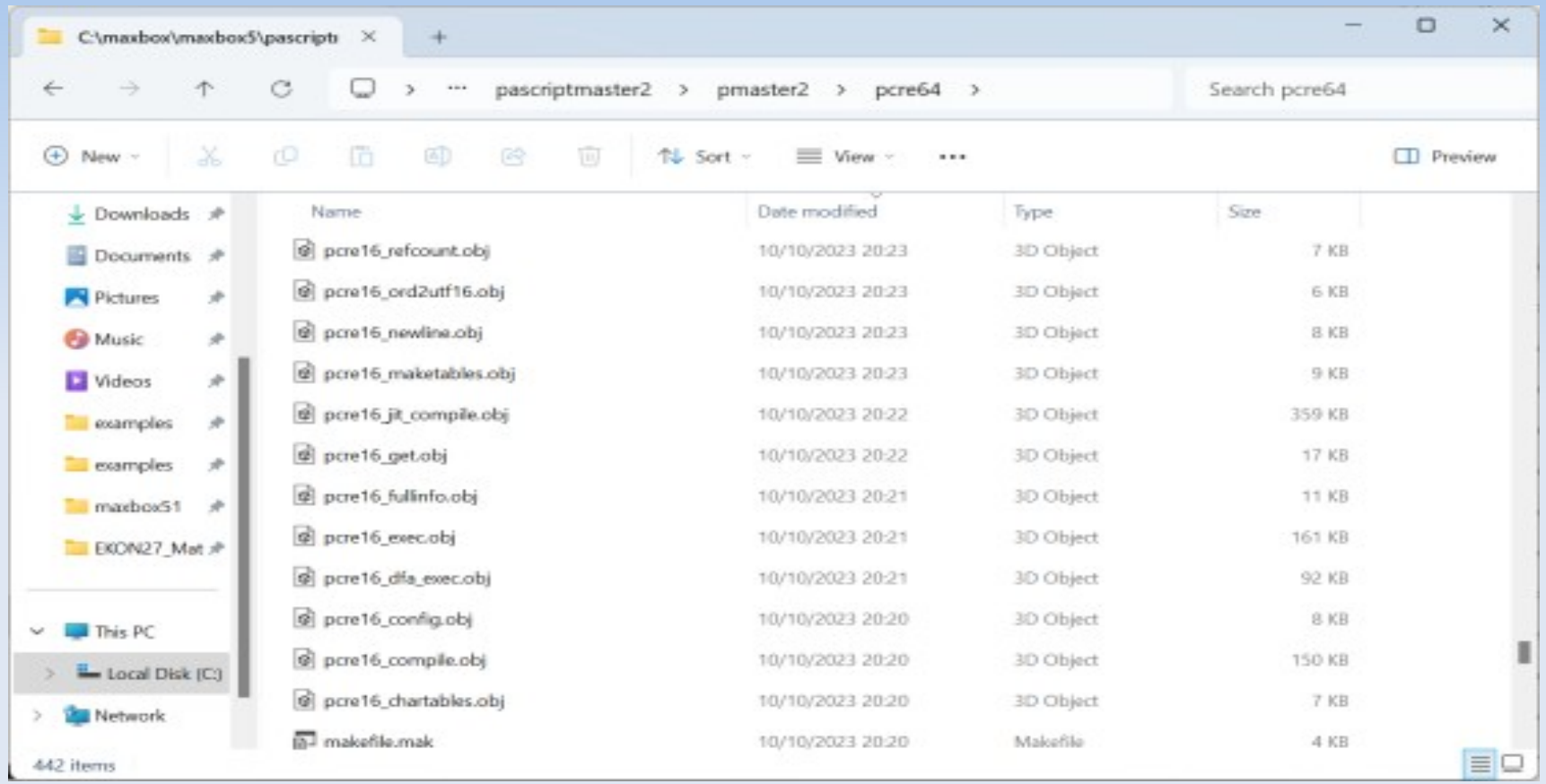
Modern Regex

Nov. 2024 \/ Max Kleiner

- TPerlRegEx with out of the box demos
- PCRE library
- TRegEx with TMatch/TMatchEvaluator
- https://www.pcre.org/original/doc/html/pcre16.html
- This session shows you various ways of using modern regex in your application.
- Python re lib
- https://regex101.com/

Compile first



- *The supplied pcrelib.dll contains PCRE 7.9, compiled with Unicode support which works with FreePascal, Lazarus, Delphi, Jupyter and maXbox.*

RegEx Research

- TPerlRegEx is a Delphi VCL wrapper around the open-source PCRE (Perl-Compatible Regular Expressions) library. It provides powerful regexp capabilities similar to those found in the Perl programming language.
- This version of TPerlRegEx is compatible with the TPerlRegEx class in the RegularExpressionsCore unit in Delphi XE.
- https://maxbox4.wordpress.com/2024/05/10/modern-regex/
- https://entwickler-konferenz.de/blog/machine-learning-mit-cai/

Cross-platform RegEx

- Use the TRegEx class from the System.RegularExpressions unit.
- This class provides methods and properties for working with regular expressions, such as Match () and Replace () for matching and replacing strings, and Captures () and Groups () for accessing matched groups.

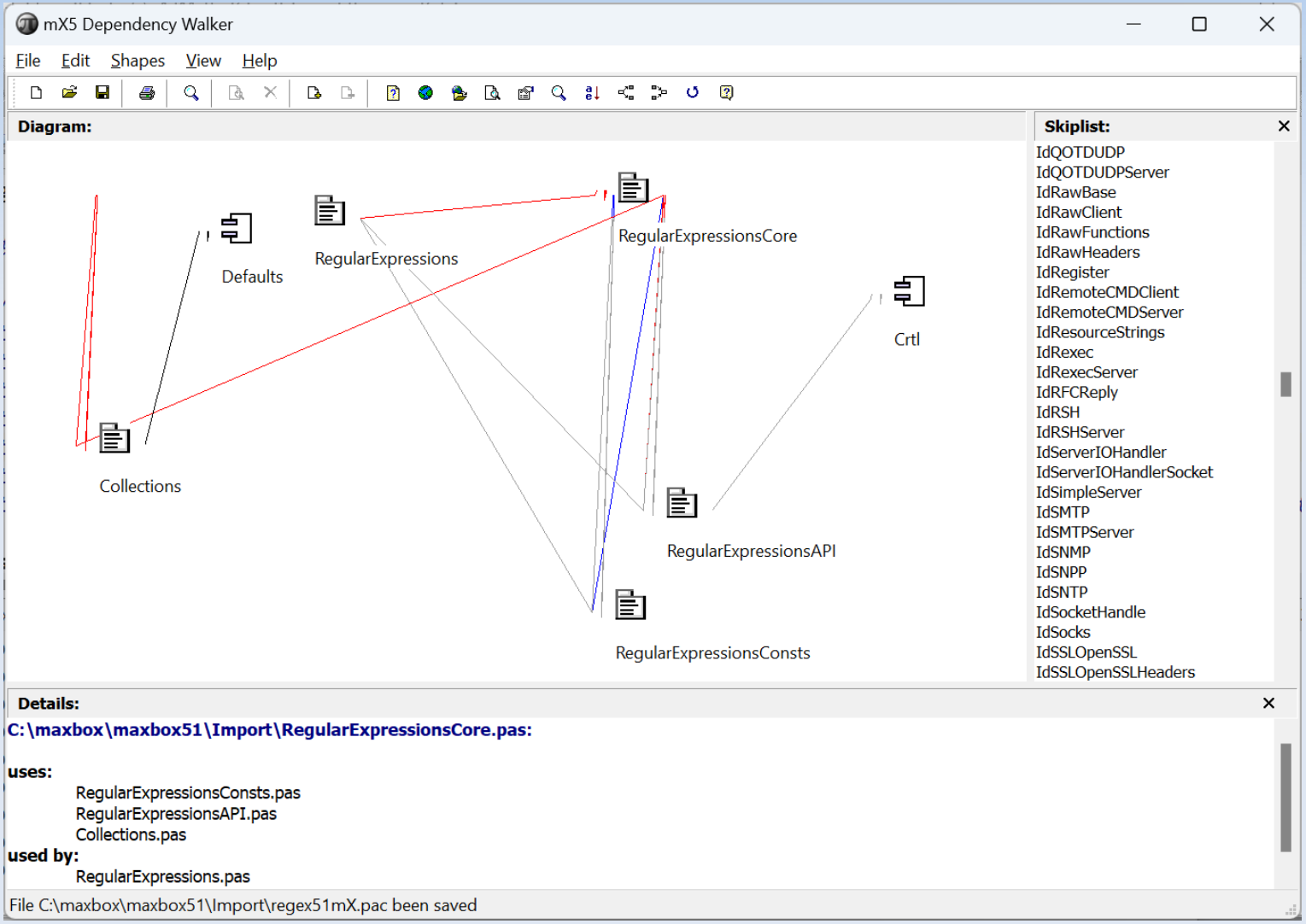
Let’s practice: maxbox51\examples\1313_regex_db12.pas

TestRegExMultiMatcher .\1317_regex_matchevaluator1.pas

https://github.com/maxkleiner/maXbox/blob/master/logisticregression2.ipynb

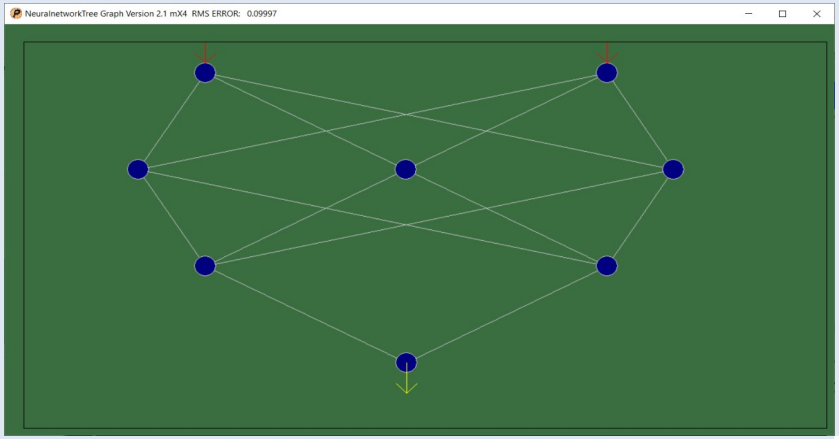
Packages View

- RegularExpression as Client uses Core (uses System.SysUtils, System.RegularExpressionsCore;)

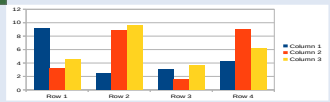


Web & RegEx = WebEx

- writeln(RegExMatch(IdWhois1.WhoIs('domain ibm.com'), '.*Registry Expiry Date.*', false));
- > Registry Expiry Date: 2025-03-20T04:00:00Z
- writeln(RegExMatch(IdWhois1.WhoIs('domain wordpress.com'), '.*Registry Expiry Date.*', false));
- > Registry Expiry Date: 2033-03-03T12:13:23Z



.\examples\1302_restcountries_API_24_mcJSON1regexEKON28.txt



Be aware of static record

Performance Object versus Static

The source for TRegEx.IsMatch(const Input, Pattern: string; Options: TRegExOptions) shows that a TRegEx is created at every invocation (which is a costly operation):

```
procedure TForm1Button1ClickRegExPerformance1(Sender: TObject);
var
  i: integer;
  t1, t2: cardinal; arex: TRegEx;
begin
  t1 := GetTickCount;
  arex:= TRegEx.Create1('')
  for i:= 1 to 100000 do
    arex.IsMatch3('abcdef', 'cd', [rroIgnoreCase]);
  t2:= GetTickCount;
  writeln('rextime1: '+IntToStr(t2-t1)+' ms');
end;
```

Conclusion: Use explicit object instance, especially in loops

Demo: https://maxbox4.wordpress.com/2024/05/10/modern-regex/

TMatch

A more modern implementation is to code with a TMatch and TMatchCollection class.

This example demonstrates the use of TMatchCollection and TGroupCollection. This example assumes that you have placed a TButton, a TEdit and a TMemo on a form.

```
// Creates and lists the match collection, the matches in that
// collection and the groups in those matches.
procedure TForm1.Button1Click(Sender: TObject);
const
  bigString = 'Look for a the strings in this strang of strungs.';
  littlestring = '(str)([iau]ng)';
var
  regex: TRegEx;
  i, j: integer;
  mygrps: TGroupCollection;
begin
  regex:= TRegEx.Create(littlestring);
  mycoll:= regex.Matches(bigString);
  Edit1.Text:= 'Count: ' + IntToStr(mycoll.Count);
  memo1.Lines.Add('First Collection: ');
  for i:= 0 to mycoll.Count-1 do begin
    memo1.Lines.Add('Match #' + IntToStr(i) + ': ' + mycoll.Item[i].Value)
    memo1.Lines.Add('Group: ' + IntToStr(i));
    mygrps:= mycoll.Item[i].Groups;
    for j:= 0 to mygrps.Count-1 do
      memo1.Lines.Add('Value: ' + mygrps.Item[j].Value);
  end;
end;
```

TMatch II

The item of a TMatchCollection returns the Match identified by index from the collection (ex. tmatches[it-1].value] below).

https://docwiki.embarcadero.com/CodeExamples/Alexandria/en/TMatchCollectionCount_(Delphi)

In general matches from a TRegEx returns all the matches present in the input string an is useful to iterate through a group or captured group (next slide):

Matches returns all the matches present in the Input string in the form of a TMatchCollection instance. If the Pattern parameter is not present the regular expression used is specified in the TRegEx constructor.

StartPos specifies the starting position to start the search. TMatchCollection has no public constructor. It is created as the return value of the Matches method. The collection is populated with one TMatch instance for each match found in the input string. The Count property is a length of the TMatchCollection set. Length specifies the substring, starting at StartPos to match with the regular expressions.

Code as script:
https://sourceforge.net/projects/maxbox/files/Examples/13_General/646_pi_evil2_64_12.TXT/download

From PI Package

- Numeric Analysis of PI Explore

```
function getMatchString2(arex, atext: string): string;
var  Match: TMatch;  tMatches: TMatchCollection;
       myenum: TMatchCollectionEnumerator;
 begin
   with TRegEx.Create1(arex)  do
   try
     it:= 0;
     { Match format search...}
     result:= result+CRLF;
     if ismatch(atext) then
         tMatches:=Matches(aText);
     writeln('captured groups: '+itoa(tmatches.count ));
       repeat
         Inc(it);
         result:= result+Format(#09'%d: %-12s',[it, tmatches[it-1].value])
         if it mod 5=0 then
           result:= result+#13#10;
       //until match(atext).success;  //MatchNext < 0;
       until it = tmatches.count;
   finally
     Free;
   end;
   WriteLn('Done REX2 - Hit NOthing to exit');
 end;
```

Big Iterator as Collection

- TMatchCollection has no public constructor. It is created as the return value of the Matches method. A collection is populated with one TMatch instance for each match found in input string.

```
regEx:= TRegEx.create('common":"[\w]',[rroNotEmpty]);
 // Execute search of TMatch
 for it:= 0 to envlist.count-1 do
   if regEx.match((envlist[it])).success then begin
     writeln(itoa(cnt)+':'+envlist[it]);
     inc(cnt)
   end;
```

langitem: 22 Schweiziska edsförbundet, swe
langitem: 23 İsviçre Konfederasyonu, tur
langitem: 24 سوئیس متحده, urd
langitem: 25 瑞士联邦 , zho

Using Groups

- System.RegularExpressions.TMatch.Groups
- *Contains a collection of groups from the most recent match with a regular expression.*
- A regular expression pattern can include subpatterns, which are defined by enclosing a portion of regex pattern in parentheses. Every such subpattern captures a subexpression or group. For ex., the regex pattern (\d{3})-(\d{2})-(\d{4}), which matches social security numbers.
  The first group consists of the first three digits and is captured by the first portion of the regular expression, (\d{3}).

```
writeln(regx1.match3('2-2321 55-99878 456-545','(\d{2})-(\d{4})').value);
```

1311_RestClientLibrary_httprequestC_EKON28.txt

Let's compile

- Contains a collection of groups from recent match with a reg expression.

maXbox5 64-bit ScriptStudio 646_pi_evil2_64_12.TXT

File Program Options View Debug Output Help

Load | Find | Replace / Refact | Go Compile! | Use Cases | Tutorial | Resource | Serial9 | maXbox5

12 646_pi_evil2_64_12.TXT

```
93   bigString = 'Look for a the strings in this strang of strungs.';
94   littlestring = '(str)([iau]ng)';
95
96 procedure TForm1Button1Click(Sender: TObject);
97 var
98   regex: TRegEx;
99   i, j: integer;
100  mygrps: TGroupCollection;
101  mycoll: TMatchCollection;
102  myenum: TMatchCollectionEnumerator;
103 begin
104  regex:= TRegEx.Create1(littlestring);
105  mycoll:= regex.Matches(bigString);
```

```
Interface List: 646_pi_evil2_64_12.TX
*******************************
 function ReadUntil(const ReadFrom, l
function StripTags2(const S: string): st
function getMatchString(arex, atext: st
 Function TimesTable2(row,col: intege
procedure TForm1Button1Click(Sender
procedure TForm1Button4Click(Sender
procedure ParseAttributes(AInputdata:
function getMatchString2(arex, atext: s
 function getMatchStringSortGroup2Re
  function getMatchStringSortGroup3Re
 procedure rex_tester_dual;
 function getMatchStringSortGroup2(ar
 function getMatchStringSortGroup3(ar
procedure RexusageExample;
Locs: 555 - code blocks: 14
```

maXbox5 C:\maxbox\maxbox51\examples\646_pi_evil2_64_12.TXT Ct:10/05/2024 16:26:38 Mem:74% Row: 93-Col: 45 s: 2974 S

```
Count: 3
First Collection:
Match #0: string
Group: 0
Value: string
Value: str
Value: ing
Match #1: strang
Group: 1
Value: strang
```

Collection Matches

- A collection of groups as the result of a match with a single regular expression. A regex pattern can include subpatterns, which are defined by enclosing a portion of the regex.

```
var mygrps: TGroupCollection;

regEx:=TRegEx.create('"common":"[\w]*(su)*',[rroNotEmpty,rrosingleline]);
   mycoll:= regex.Matches(envlist.text);
   writeln('Count: ' + IntToStr(mycoll.Count));
   // Execute search3 ot TMatchCollection
   for it:= 0 to mycoll.count-1 do begin
      writeln(itoa(it)+':'+(mycoll.Item[it].Value));
      mygrps:= mycoll.Item[it].Groups;
      for j := 0 to mygrps.Count-1 do
        writeln('Value: ' + mygrps.Item[j].Value);
   end;
regex.Free;
envlist.Free;    //*)

```

The Main Python

```
procedure PyCodeREgEx(tex: string);
begin
  with TPythonEngine.Create(Nil) do begin
  try
    loadDLL;
    //autofinalize:= false;
    ExecString('import re, json');
    execstr('txt = "The rain in Explain"');
    //if it starts with "The" and ends with "plain":
    Println('regex test: '+EvalStr('re.search("^The.*plain$", txt)'));
    execstr('x = re.search("^The.*plain$", txt)');
    Println('regex test: '+EvalStr('type(x)'));
    Println('regex test: '+EvalStr('bool(x)'));
   // Println(EvalStr('decode_and_print_json('+JSONDATA+')'));
  except
    raiseError;
  finally
    unloadDLL;
    free;
  end;
 end;
end;
```

re module

Python has a built-in package called re, which can be used to work with Regular Expressions as re.match() or re.search().

RegEx Functions

The `re` module offers a set of functions that allows us to search a string for a match:

Function	Description
findall	Returns a list containing all matches
search	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

- https://docs.python.org/3/library/re.html

Demo FindFiles()

```
procedure TForm1FormCreate(Sender: TObject);
var k,t: integer;
  items: TStringList;
begin
  items:= TStringList.create;
  for k:= 0 to 9 do
    StringGrid1.Cells[0, k+1]:= cs10Labels[k];

  //FindAllFiles(ComboBox1.Items, 'csdata');
  FindFiles(exepath+'data', '*.bmp',items);
  writeln(items.text);
  for t:= 1 to items.count-1 do
     ComboBox1.Items.add(items[t]);
  if ComboBox1.Items.Count > 0 then begin
    ComboBox1.text:= ComboBox1.Items[0];
    if FileExists(ComboBox1.text) then begin
      Image1.Picture.LoadFromFile(ComboBox1.text);
      Image2.Picture.LoadFromFile(ComboBox1.text);
      label1.Caption:= extractfilename(ComboBox1.text);
    end;
  end;
end;
```

Compare Delphi Python

```
procedure PyCodeREgEx(tex: string);
begin
  with TPythonEngine.Create(Nil) do begin
  try
    loadDLL;
    //autofinalize:= false;
    ExecString('import re, json');
    execstr('targetstr = "Emma''s luck numbers are 251, 761, and 23
    execstr('pattern = re.compile("\d{3}")');
    execstr('matches = pattern.findall(targetstr)');
    Println('py regex test: '+EvalStr('matches'));
  except
    raiseError;
  finally
    unloadDLL;
    free;
  end;
 end;
end;
```

```
writeln('show matchall/findall in python:');
 with TregEx.create1('\d{3}') do begin
   matchg:= Match('Emma''s luck numbers are 251, 761, and 231 ')
   repeat
     write('[ '+ matchg.value+' ]');
     matchg:= matchg.NextMatch;
   until not matchg.success
   free;
 end;
```

Unicode Group Samples

// {$APPTYPE CONSOLE} ☮ ✞ π 📌

- writeln(regx.ReplaceAll('\u0418\u0443, \u0427\u0436\u044d\u0446\u0437\u044f\u043d',
  '\\u([0-9a-f]{0,4})','$$+', [rroIgnoreCase,rroSingleLine]));

UC_teststr:= 'Düsseldorf, Köln, 北京市 , إسرائيل, دوسلدورف , Αλφα !@#$';

myEval: TMatchEvaluator;

```
myeval:= @EvaluatorU;
  Writeln(regx.Replace7('\u0418\u0443, \u0427\u0436\u044d\u0446\u0437\u044f\u043d',
                '\\u([0-9a-f]{4})', myeval, [rroIgnoreCase]));
```

mycoll:= regx1.matches2('match non-english words like können or móc zu Çin', '(?s)(.[^\x00-\x7F]\b)+')

https://www.regexpal.com/

https://github.com/maxkleiner/maXbox/blob/master/objectdetector3.ipynb

Conclusion

- Internally Delphi uses class TPerlRegEx and has such methods for groups and collections.
- Number of matched groups stored in the Groups array. E.g. when the regex "(a)|(b)" matches "a", GroupCount will be 1. When the same regex matches "b", GroupCount will be 2.
- The static TMatch record or class as instance provides several properties with details about the match. Success indicates if a match was found.
- You can use a numeric index to Item[] for numbered capturing groups, or a string index for named capturing groups thanks to variants!
  > whatGotMatched:= Match.Groups['MatchName'].Value;

Method: Design Regex with a Online Site like regex101.com
Model: Object **Regex Pattern** + Subject Data
Metric: Test with generic data and community pattern

https://raw.githack.com/breitsch2/maXbox4/master/assets/graph3.html

Thanks for coming!

Materials:

https://maxbox4.wordpress.com/2024/05/10/modern-regex/

https://maxbox4.wordpress.com/2024/06/20/ekon-28/

https://medium.com/@maxkleiner1/modern-regex-d9d3450fbd36

https://maxbox5.wordpress.com/2024/07/22/ekon-28/