9. Ders - Missin Value - Part 2 - REGEX

24 Eylül 2025 Çarşamba 23:08

Regex (Düzenli İfadeler), metin içinde desen eşleştirme, arama, değiştirme yapmak için kullanılan güçlü yöntemdir. Veri temizleme, etiket standardizasyonu, pattern çıkarımı gibi işlemlerde çok kullanılır.

- **import re**

◇ Kullanım amaçları kapsamında şunları söyleyebiliriz :

- Tekrarlı işlemlerde vakit kazanmak,
- Dosya adlarını topluca değiştirmek,
- Kullanıcının girdiği girdileri denetlemek,
- Çok sayıda ve karmaşık haldeki verilerden istenen bilgiye hızlıca ulaşmak,
- Veri tabanı kullanımını kolay hale getirmek, metin araması yapmak, veri kümelerinden hızlı sonuçlar alabilmek,
- İstenilen diziyi kolayca bulmak ve üzerinde istenilen düzeltmeleri yapmak,
- Daha kolay, hızlı ve düzenli kod yazmak için Regex kullanılır.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

REGEX SYNTAX

- . **(Nokta):** Herhangi bir karakteri temsil eder.
- * **(Yıldız):** Önceki karakterin sıfır veya daha fazla tekrarını temsil eder.
- + **(Artı):** Önceki karakterin bir veya daha fazla tekrarını temsil eder.
- ? **(Soru işareti):** Önceki karakterin sıfır veya bir kez tekrarını temsil eder.
- [ ] **(Köşeli parantezler):** Belirli karakterler kümesini temsil eder. Örneğin, [aeiou] tüm sesli harfleri temsil eder.
- [^] **(Ters köşeli parantezler):** Belirli karakterlerin kümesinin dışındaki karakterleri temsil eder.
- \b: Kelime sınırını ifade eder.
- \d: Sayı karakter kümesini ifade eder.
- \D: Sayı olmayan karakter kümesini ifade eder.
- \s: Boşluk karakterlerini ifade eder, tam tersi için \S kullanılır.
- \w: Rakam, küçük harf, büyük harf ve "_" karakterlerinin tümünü ifade eder.
- \t: Sekme boşluk(tab)
- \r: Satır Kesme (Break)
- {n}: Bir önceki karakterin tam olarak n kez tekrarını belirtmek için kullanılır.
- $: Bir satırın sonunu belirtir.
- | : Veya. Önceki koşul ya da sonraki koşuldan birisini çalıştırırsa devam eder.

- [a-Z] çalışmaz, çünkü ascii tablosunda **Z** karakteri **a**'dan önce gelir.
- [A-z] çalışır, ancak arada başka karakterleri de kapsayacağı için tercih edilmez.
- [ase?,] sadece bu 5 karakterden oluşan bir karakter kümesidir.
- [abc]: a, b veya c karakterlerinden herhangi biriyle eşleşir.
- (abc): Parantez içinde belirtilen karakter dizisini bir grup olarak eşleştirmek için kullanılır.
- [ ] ^ \ kaçış karakteri olmadan karakter kümesine dahil edilemez. Çünkü [ ] zaten karakter kümesini ifade eder. ^ ise değildir (tersi) anlamına gelir.

Regex Kapsamında Temel 3 Metot

1. **Eşleştirme (Matching):**

```
string input = "Mataramda Tuzlu Su";
string pattern = "Tuz";
bool isMatch = Regex.IsMatch(input, pattern); //True
```

Bir regular ifadenin bir metin içindeki varlığını bulmaya ve eşleşen kısımları belirlemeye yönelik bir işlemdir. **Regex.IsMatch**() yöntemi, bir metnin bir regular ifade ile eşleşip eşleşmediğini kontrol eder. Eşleştirme işlemi, metnin tamamında veya belirli bir bölgesinde gerçekleştirilebilir.

2. **Yakalama (Capturing):**

```
string input = "Abdullah Özgün";
string pattern = @"(\w+)\s(\w+)";
Match match = Regex.Match(input, pattern);
if (match.Success)
{ string firstName = match.Groups[1].Value;
string lastName = match.Groups[2].Value;
}
```

Bir regular ifadenin belirli bir kısmını ayırmak veya yakalamak için kullanılır. Parantez içinde belirtilen kısımlar, gruplar olarak adlandırılır ve eşleşen metin içindeki bu gruplar ayrı ayrı erişilebilir hale gelir. Yakalama, **Match.Groups** özelliği kullanılarak gerçekleştirilir.

3. **Replacing (Değiştirme):**

```
string input = "Nereye gittiğini bilmiyorsan, hangi yoldan gittiğinin hicbir önemi yok";
string pattern = @"\b\w{6}\b"
MatchCollection matches = Regex.Matches(input, pattern); foreach (Match match in matches)
{
Console.WriteLine($"{match.Value} \t"); //Nereye yoldan hicbir
}
```

Değiştirme işlemi, bir metindeki belirli desenleri başka bir metinle değiştirmek için kullanılır. **Regex.Replace()** yöntemi, bir metindeki tüm eşleşmeleri değiştirir ve değiştirilmiş metni döndürür.

Birkaç örnek üzerinden örneklendirmeler yapalım :

Senaryo 1 : Bir form dolduruyorsun ve kullanıcının girdiği telefon numarasının doğru formatta olup olmadığını kontrol etmek istiyorsun.

```python
import re

text = "İletişim: 0532-456-7890"
pattern = r"\d{4}-\d{3}-\d{4}"   # 4 rakam - 3 rakam - 4 rakam

print(re.findall(pattern, text))  # ['0532-456-7890']
```

Benzer formatın farklı bir koşul ile oluşturulmuş patterni şu şekilde olabilir :

```python
import re

text = "İletişim: 0532-456-7890, diğer: 0212-123-4567, cep: 0555-111-2233"
pattern = r"05\d{2}-\d{3}-\d{4}"   # 05 + 2 rakam + - + 3 rakam + - + 4 rakam

print(re.findall(pattern, text))
# Çıktı: ['0532-456-7890', '0555-111-2233']
```

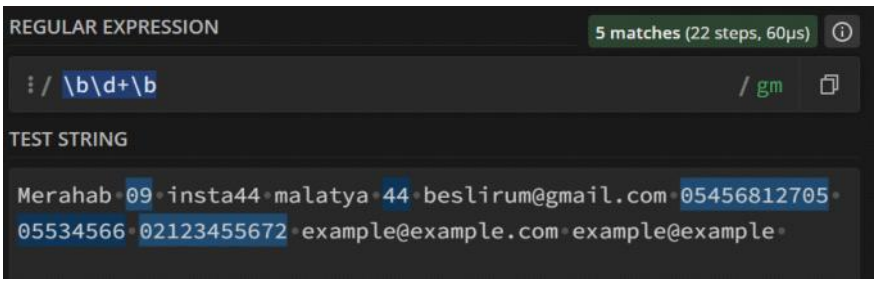
Senaryo 2 : Bir metinde geçen tüm sayıları alalım.

```python
import re

text = "Sipariş numarası 345, toplam fiyat 1200 TL, kargo 25 TL."
pattern = r"\d+"

print(re.findall(pattern, text))
# ['345', '1200', '25']
```

Bu kodda, sayıların hepsini alıyor. Peki sadece sayı almak istesek, string içinde yer alan sayıları dahil etmeyecek şekilde bütün sayısal verileri elde etmek istersek şu şekilde bir regex düzeni oluşturmamız gerekir:



(ÖRNEĞİMİ https://regex101.com/ SİTESİ ÜZERİNDEN YAPTIM.)

Senaryo 3: Bir yazıda geçen internet adreslerini bulalım.

```python
import re

text = "Kaynaklar: https://medium.com ve http://github.com"
pattern = r"https?://[a-zA-Z0-9./]+"

print(re.findall(pattern, text))
# ['https://medium.com', 'http://github.com']
```

Aşağıdaki metin için bir çalışma gerçekleştirdim :

Metin - >

Merhaba, ben Ahmet Yılmaz.
Mail adresim: ahmet.yilmaz@example.com
Alternatif adres: ayilmaz123@gmail.com
Telefonum: 0532-456-7890, ofis: 0212-123-4567
Web: https://ahmetyilmaz.com veya http://company.org
Bugünkü sipariş numarası: 34567, fiyatı: 1200 TL, kargo: 25 TL
Referans kodu: abc123xyz

1. Telefon numaraları -> **\d{4}-\d{3}-\d{4}**
2. Mail adresleri -> **\w+@\w+.com**

3. Web adresleri-> **http....\w+.\w+**
4. Sadece sayılar -> **\b\d+\d**
5. Büyük harfle başlayan isimler -> **[A-Z-ÇİÖŞÜ]\w+**
6. TL fiyatları -> **\b\d+\s?TL**