**REGEX (Regular Expressions)**

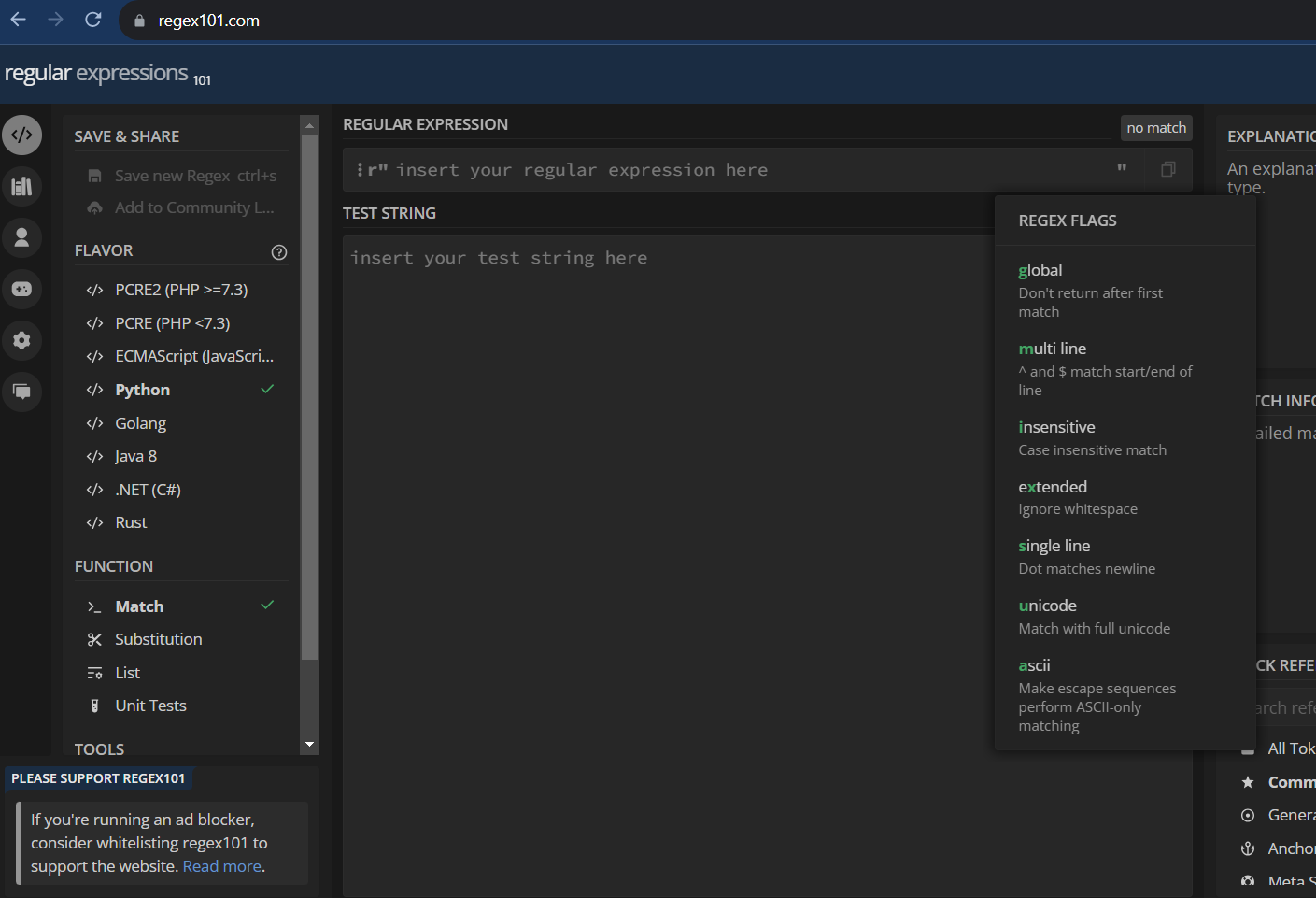
**عبارت های منظم و با قاعده (ریجکس)**

**regex ها[[1]](#footnote-1)برای هر برنامه نویسی الزامی هستش دونستنش(مخصوصا اگه بک-اند کار هستید)، و کاری که این ماژول انجام میده به صورت خلاصه این هست که روش ها و ابزار هایی برای ما فراهم میکنه که بتونیم از متن ها اطلاعات استخراج کنیم.**

**یه کاربردش مثلا میتونه این باشه که ببینه چیزهایی که شما فیلد ایمیل وارد کردید درست هست یا نه.یا مثلا بیاید هرچی عبارت "Fuck" هستش رو پیدا کنید توی یه متن و با "\*\*\*\*" جایگزین و ریپلایس کنید. و یا حتی توی وب اسکراپینگ که مثلا فقط فایل هایی با پسوند "mp4" رو دانلود کن و ......**

**این مبحث توی همه زبان ها هستش و همشون مثلا کاراکتر ؟ رو دارند ولی شاید کاربردش فرق داشته باشه.**

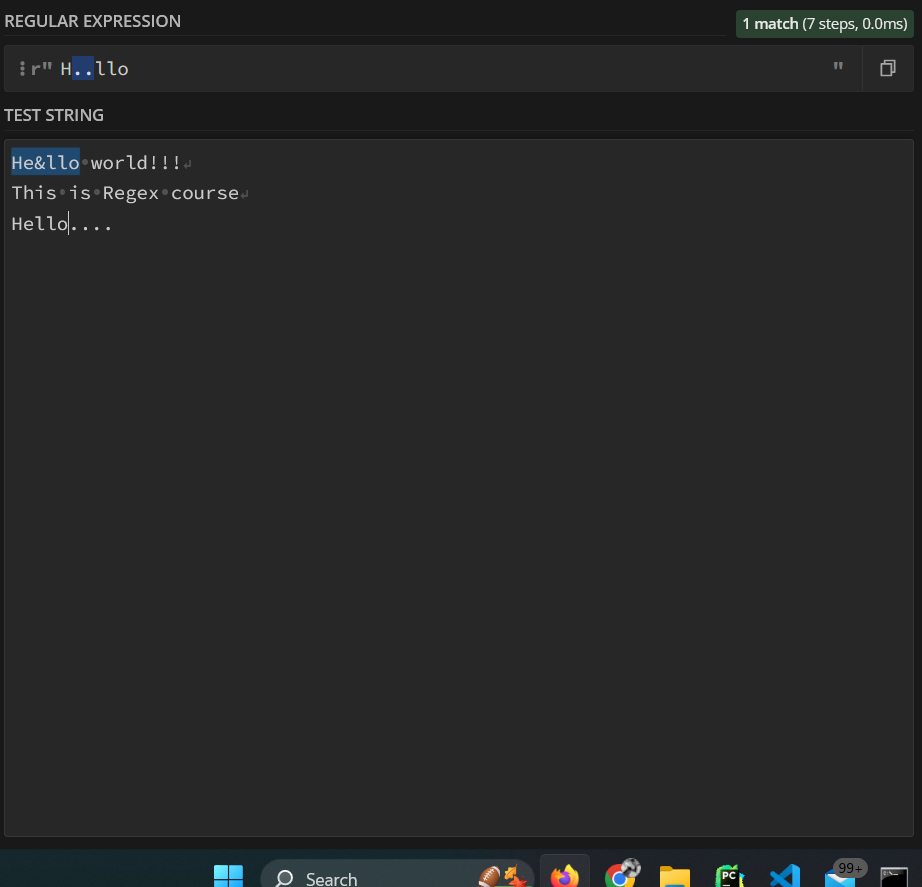
**برای تست و کار با ریجکس هاتون سایت regex101 گزینه ی خوبیه.(توی قسمت فلگ ها بهتره که همرو خاموش کنید.)**

****

*تصویر 1: محیط تست و آموزش*

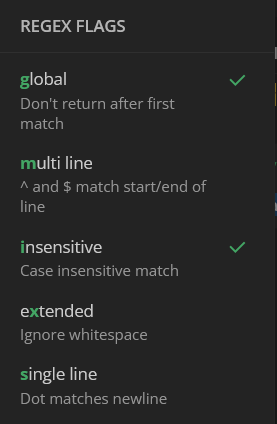
**شروع و معرفی چندین کاراکتر :**

* **Dot (.) :** در حالت پیش فرض هر کاراکتری رو قبول میکنه(حتما باید یک کاراکتر باشه) به غیر از خط جدید. ( ؟ و # و . و Tو غیره).

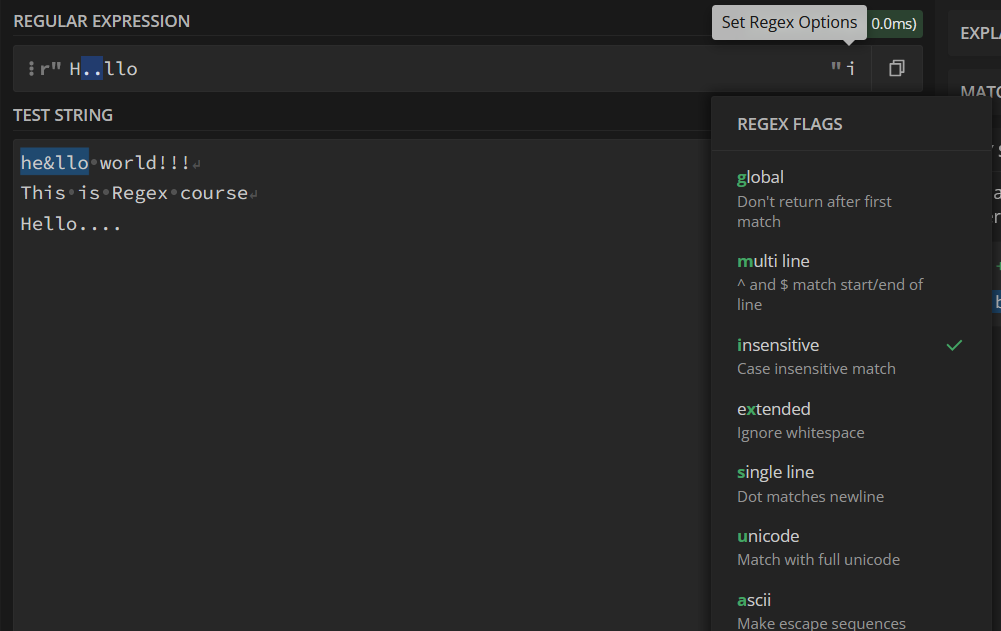
****

*تصویر 2: استفاده از " . "*

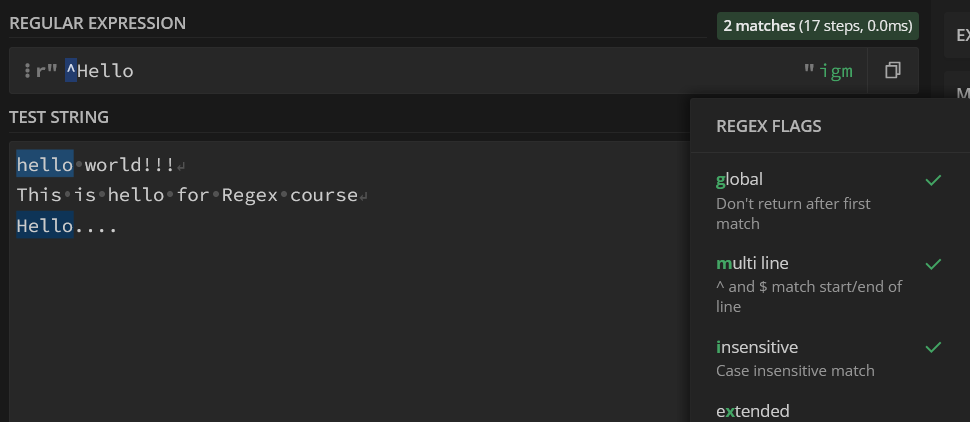
با استفاده از DOTALL میتونیم تغییراتی توش اعمال کنیم مثله اینکه enter رو هم قبول کنه که اختصارا بهش s هم میگن و توی این عکس اگه فلگ single line رو بزنی همون کار رو میکنه.

******

***نکته :*** *با استفاده از فلگ ها ما میتونیم یه سری تغییراتی روی رفتار ریجکس ها اعمال کنیم . مثلا : ریجکس ها به حروف بزرگ و کوچک حساسند و ما میتونیم با استفاده از* IGNORECASE *این رو غیر فعال کنیم. یا حتی با استفاده از* global ( g) میتونیم همه ی hello توی متن را پیدا کنیم نه فقط اولی.

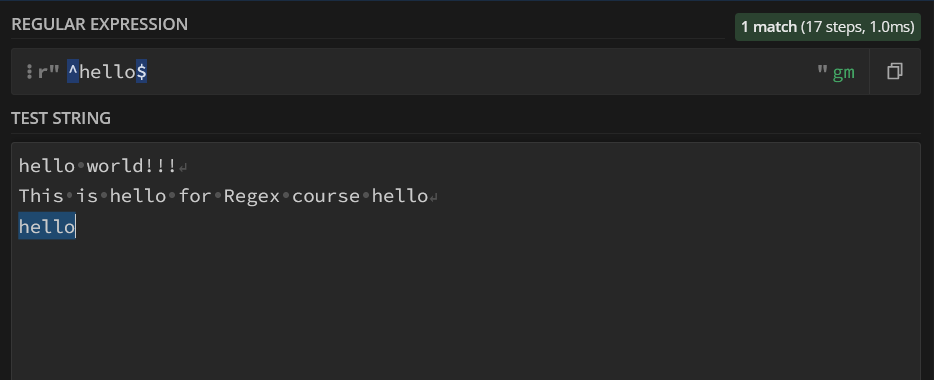
**** *تصویر 3: استفاده از " . “*

* **Caret(^) :** ابتدای یک استرینگ رو برای ما مطابقت میده.(نه وسطای جمله و آخرش) و با استفاده از Multilineمیتوانیم چندین خط را بررسی کنیم.



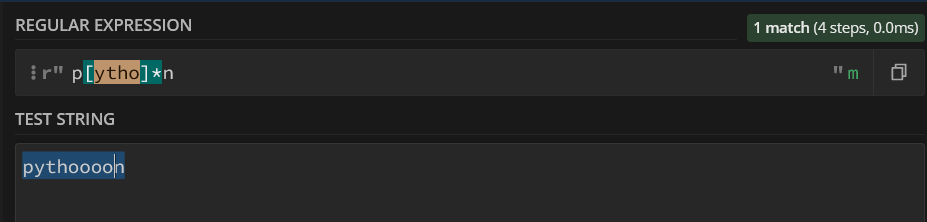
*تصویر 4: استفاده از "^ "*

* **Dollar sign($) :** دقیقا بر عکس " ^ " هستش و آخره خط رو برمیگردونه.

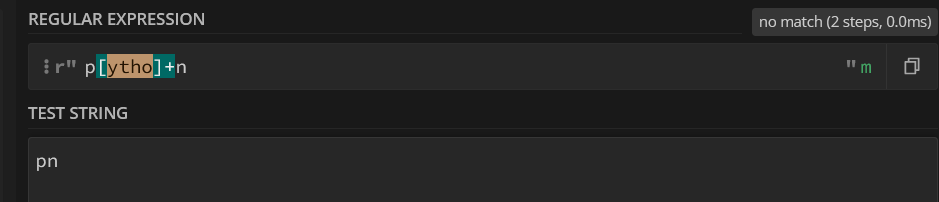


*تصویر 5 : اینجا بخاطر ^ باید کلمه ای پیدا بشه که اولش h باشه و آخرش هم o*

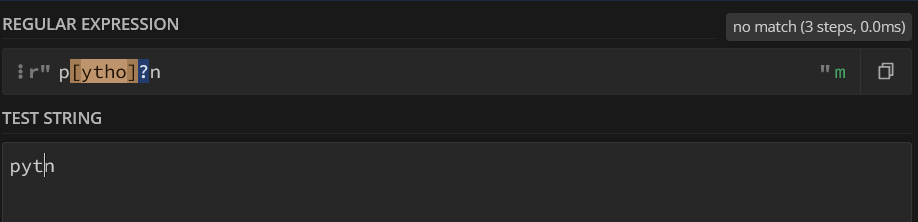
* **Asterisk(\*) :** یعنی پیدا کردن صفر یا بیشتر کاراکتر مثلا توی شکل زیر گفتم که اولش p و آخرش n باشه، وسطش هم میتونه هیچی نباشه و میتونه از اون حروفی که قرار دادم هرچقد مه میخواد تکرار بشه.(Zero or more)



و علامت + هم برای وقتیه که حتما یکی از اون کاراکتر ها توش باشه.(One or more )



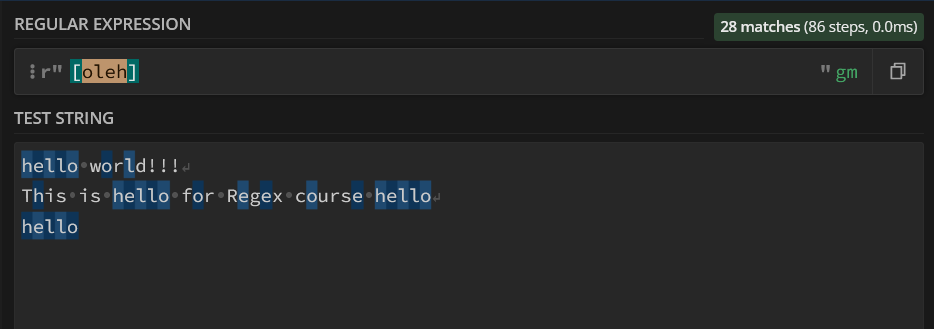
و علامت ؟ هم یعنی کاراکتر یا باید یکی باشه یا اصلا نباشه. (Zero or one)



برای اینکه به صورت خام استفاده کنی میتونی قبلشون یه دونه" \" بزاری.

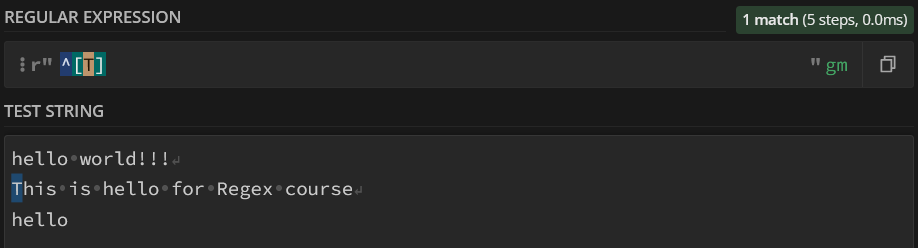
**براکت ها :**

به چند حالت مختلف میتونیم نشونش بدیم انتظار چه کاراکتر هایی رو داریم که وجود داشته باشه ، میتونیم هم به صورت بازه ای (range) مشخص کنیم.

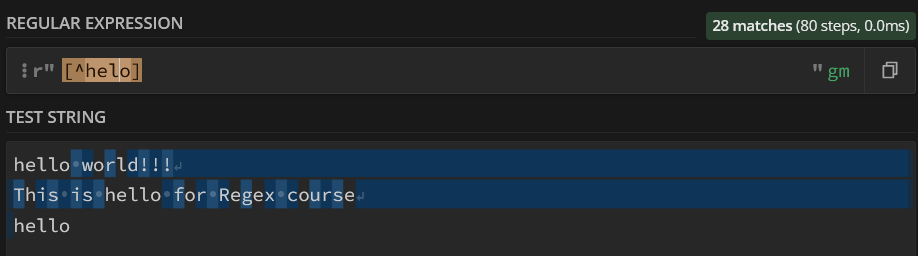
****

میتونیم ^ رو با براکت ترکیب کنیم که حالات مهم زیر ممکنه بوجود بیاد:

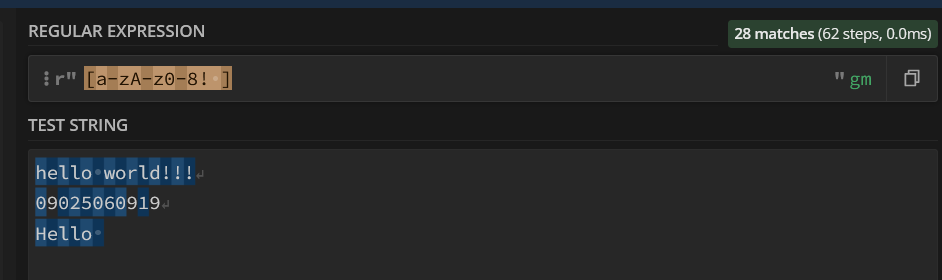
1 - ^ رو قبل از براکت بزاریم : یعنی حروفی رو بیار که اول جمله باشند(میتونستی چنتا حرف هم بزاری)



2- ^ رو داخل براکت (قبل از حروف) بزاریم ، که یعنی این حروف رو در نظر نگیر کلا. (except)



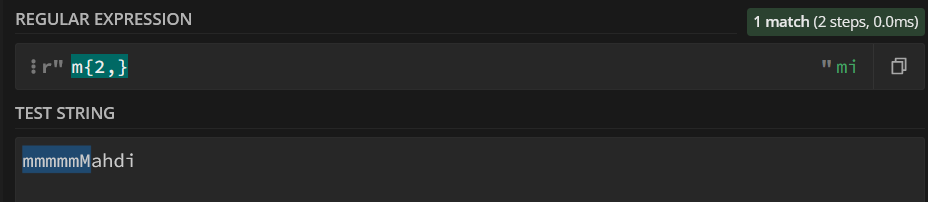
برای کار کردن با بازه هاش هم از "-" استفاده میکنیم. مثلا تو شکل پایین بهش گفتم که از a تا zو از 0 تا 8 رو به همراه ! و حتی فاصله و خط بعد رو (\n ,space) بیاره برامون. (برای بقیه کاراکتر ها مثلا # و \* و غیره باید خودشون رو بنویسیم طبیعتا)



* **Curly bracket {} :**

براکت های فرفری (مثل خودم) یا همون آکولاد ها میتونی تعداد تکرار هارو مشخص بکنی.(دوتا عدد بفرستی در اصل براش بازه تعریف کردی).

میتونی عدد اول رو هم قرار ندی که از صفر شروع بشه و برعکس : { , 9 } یا { , 1 }

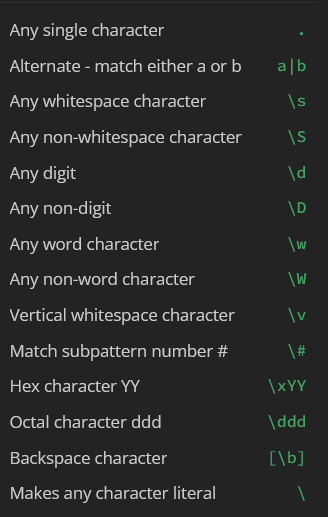


* **: Sequences” \ ”**

برای مشخص کردن جملات هستش و یکم توضیحشو نمیدونم چجوری میشه گفت ولی مثلا بالاتر گفتیم اعداد 0-9 رو بیاره برامون(]0-9[) ولی کافیه اینجا بگی \d تا همه اعداد رو بیاره و \Dهم یعنی همه چیو بیاره جز اعداد . چنتا از مهم هاشو میگم این پایین :

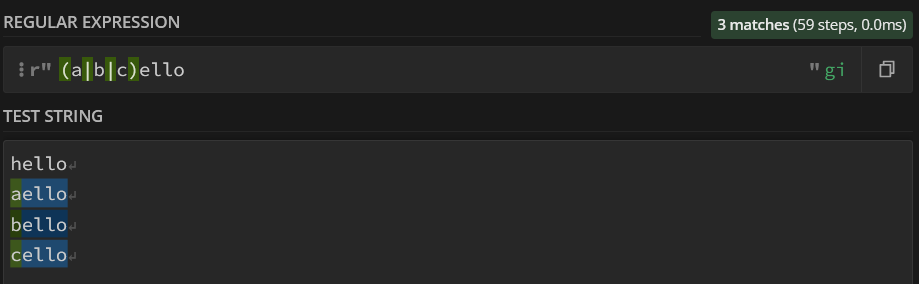
\b هم یعنی قبل از فلان کاراکترم نمیخام چیزی باشه(مرز بین حروف رو مشخص میکنه) : \bhello قبل از hello هیچی نباشههههه. به جز کاراکتر های خاص (\*&#$) و فاصله و enter

| به معنی یا هستش، مثلا a|b|c کلماتی رو که اولشون با یکی ازa , b , c شروع شده باشه رو میاره.



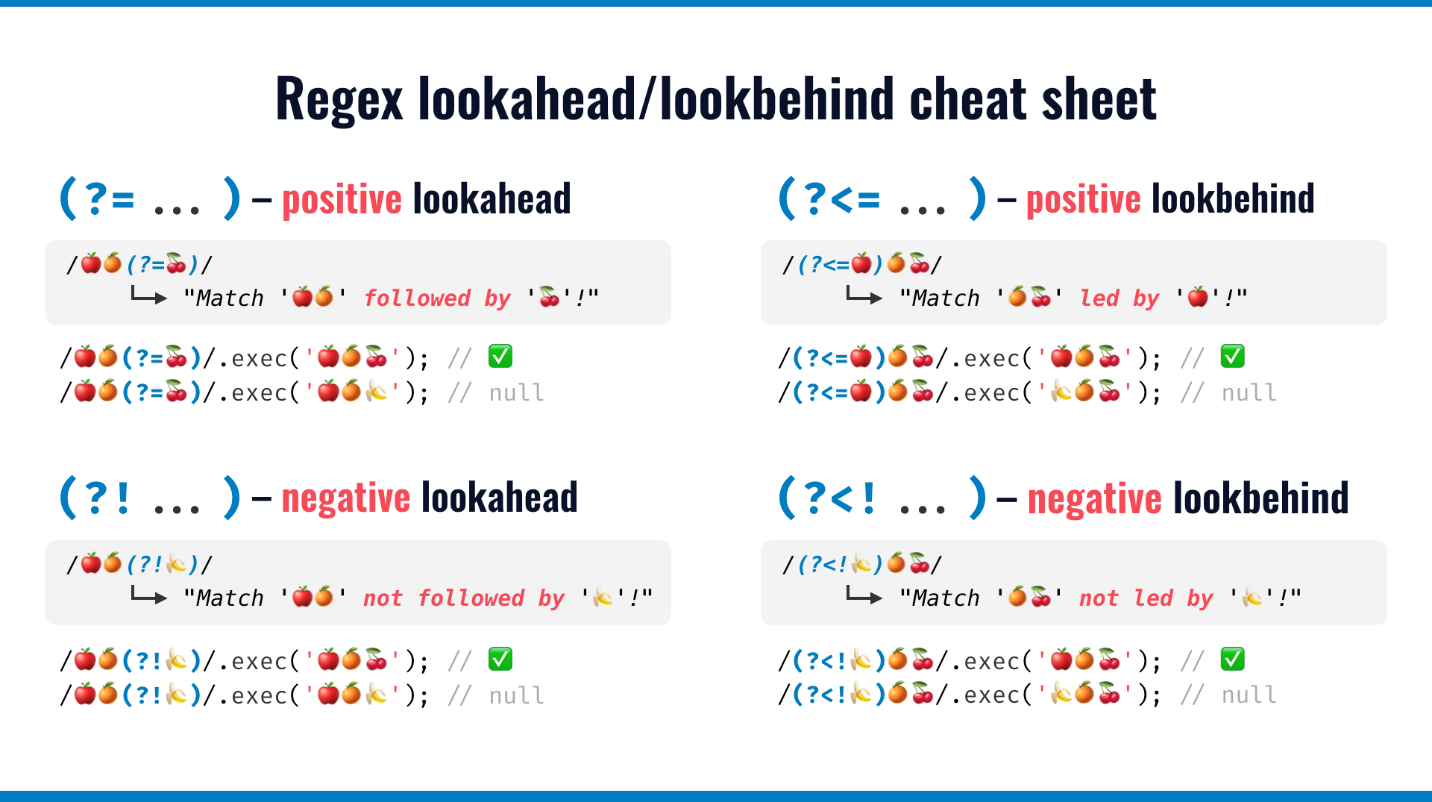
* **: Parentheses()**

میتونیم دستوراتمون رو گروه بندی کنیم و مثلا چنتا دستور رو بهش بدیم .

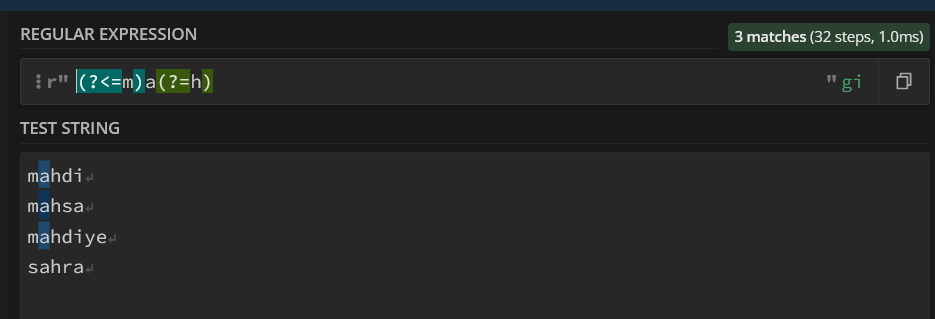


* **Lookaround :**

فرض کنید میخاید تمام کاراکتر هایی که قبلشون یا بعدشون فلان حرف وجود را پیدا کنید.(دقت کنید که فقط اون حرف رو میخاید نه حروف قبلو بعد، مثلا تمام a هایی که قبلش m بوده ). اینجا میایم از اینها استفاده میکنیم که چهارتا حالت داره:



معمولا زیاد استفاده هم نمیشند ولی گفتم بهتون یه مثال هم خودم میزنم :



**نحوه استفاده از Regex در پایتون :**

**ما با ایمپورت کردن ماژول میتونیم از استفاده کنیم. مثل بقیه ماژول های پایتون متود ها و توابع های خاص خودش رو داره که تو باید از اونا پیروی کنی.**

* **.compile()**

مثلا این فانکشن از ما دوتا مقدار میگیره ، یکیpattern و flags که اولیش دقیقا همون چیزهایی هستش که ما توی تکستمون دنبالش هستیم و فلگ هارو هم که بالاتر گفتیم چجوریه.

* **.match()**

مثلا این فانکشن هم اولین آرگومانی که میگیره همون پترن بالاییه و دومیش اون رشته ای هستش که میخای روش سرچ بزنی.

مثال زیر رو با دقت بررسی کنیم میفهمیم که کلا نحوه کار به چه شکلی هستش.

نکته : برای اینکه فقط کلمه رو تو خروجی ببینی میتونی از سینتکس استفاده کنی : print(res.match)

نکته : توی پایتون خواستی با ریجکس ها کار کنی قبل از اون پترنت یه “r” بزار، چونکه بهش کمک میکنی راحت تر اسکیپ کنه و بهتره. توی مثال من یادم رفته بزارم ولی تو یادت نره....



دوتا فانکشن match و search نمیتونند چنتا رو برگردونند برای همین میتونیم از فانکشن findallاستفاده کنیم ولی بدیش اینه که به صورت لیستی برمیگردونه و فقط کلمات رو میده بدون ایندکس هاش. برای حل این مشکل از فانکشن استفاده میکنیم (چون آبجکت میسازه باید این ایتربل فور بزنی یا تبدیل به لیستش کنی)



**مثال تشخیص ایمیل با استفاده دیجکس ها در پایتون :**

****

1. برای صرفه جویی در وقت عبارت re به عنوان ریجکس در متن میباشد. [↑](#footnote-ref-1)