**lambda :**

در پایتون ، یک تابع ناشناس ممکن است تابعی باشد که بدون شهرت تعریف شده باشد. هنگام تعریف توابع عادی، ما از کلمه کلیدی def در پایتون استفاده می کنیم، اما هنگام تعریف توابع ناشناس از کلمه کلیدی lambda استفاده می کنیم. لامبدا روشی سریع برای ایجاد تابع است. عبارات لامبدا باید به ساده ترین شکل ممکن و به دور از هرگونه پیچیدگی نوشته شوند

## **# چطور از lambda استفاده کنیم؟**

ساختار نگارش کلی عبارات لامبدا به شکل زیر است:

lambda (arguments): manipulation(arguments)

ابتدا با استفاده از کلمه کلیدی lambda مشخص میکنید که قراره یک لامبدا داشته باشید. بعد از اون آرگومان هایی که قرار است عبارت لامبدا داشته باشه رو مینویسید. و در آخر کاری قراره روی آرگومان ها انجام بشه رو مینویسید. دقت کنید که عبارت lambda هر تعدادی که بخواهید میتواند آرگومان داشته باشد اما فقط یک manipulation خواهد داشت.

### **+ یک مثال از عبارت لامبدا**

به مثال زیر دقت کنید:

add = lambda x, y: x + y

print( add(4, 6) )

در مثال بالا یک عبارت لامبدا ساختیم که دو مقدار x و y رو میگیره و اونها رو با هم جمع میکنه. و در آخر لامبدا رو داخل یک متغیر ذخیره میکنیم تا بتونیم بعدا ازش استفاده کنیم. در خط آخر هم لامبدا رو صدا زدیم و دو عدد رو بهش فرستادیم. اگه کد بالا رو اجرا کنید مقدار ۱۰ رو به شما برمیگردونه. دقت کنید که لامبدا به صورت اتوماتیک مقدار نهایی رو return میکنه.

+ چرا از عبارت لامبدا استفاده کنیم؟

همانطور که در قسمت قبل مشاهده کردید، پایتون با لامبدا به شکل یکسان با فانکشن‌های معمولی برخورد میکند. به نوعی، lambda شکلی جمع و جور برای نوشتن توابع ارائه می دهد که یک عبارت را باز می گرداند. با این حال، شما باید بدانید که استفاده از لامبدا چه زمانی مناسب است و چه زمانی از آنها اجتناب کنید.

یکی از مهمترین موارد استفاده رایج برای lambda، برنامه نویسی فانکشنال است زیرا پایتون از پارادایم (یا سبک) برنامه نویسی که به عنوان برنامه نویسی فانکشنال شناخته می شود پشتیبانی می کند.

لامبدا به شما این امکان را می دهد که یک تابع را به عنوان پارامتر به یک تابع دیگر (به عنوان مثال، در map، filter و غیره) ارائه دهید. در چنین مواردی، استفاده از lambda به لطف ایجاد یک تابع یک بارمصرف، مفید است.

در محیط production، هرگز نباید توابع lambda پیچیده را بنویسید، زیرا رمزگشایی برای کد نویسان که کد شما را حفظ می کنند بسیار دشوار است. اگر خودتان را در حال ساختن عبارات پیچیده تک خطی کشف کردید، بهتر است با def یک فانکشن کامل بنویسید.

## **# استفاده از lambda با filter**

تابع filter برای فیلتر کردن برخی عناصر خاص از یک دنباله استفاده می شود. دنباله مورد استفاده در این تابع یک [آبجکت ‌iterable](https://www.mongard.ir/one_part/53/creating-iterable-objects-python/) مانند لیست ها ، مجموعه ها ، چندتایی ها و غیره است.

sequences = [10,2,8,7,5,4,3,11,0, 1]

filtered\_answer = filter (lambda x: x > 6, sequences)

print(list(filtered\_answer))

# OUTPUT

# [10, 8, 7, 11]

# استفاده از lambda با map

تابع map برای استفاده از یک عملیات خاص برای هر عنصر در یک دنباله استفاده می شود.

sequences = [10,2,8,7,5,4,11]

squared\_result = map (lambda x: x\*x, sequences)

print(list(squared\_result))

# OUTPUT

# [100, 4, 64, 49, 25, 16, 121]

=================================================

**\_\_str\_\_ vs \_\_repr\_\_ :**

این دو متد برای زمانی هستش که شما میخواید یک آبجکت رو چاپ کنید:

class Person:

def \_\_init\_\_(self, name):

self.name = name

p1 = Person('amir')

print(p1)

زمانی که کد بالا رو اجرا میکنیم به شکل زیر نتیجه رو به ما نشون میده:

<\_\_main\_\_.Person object at 0x7f2772b0cf50>

پاسخی که پایتون برای ما آورده آنچنان مطلوب نیستش و نمیشه چیز زیادی ازش فهمید. برای تغییر دادن این رفتار پایتون از متد str استفاده میکنیم:

class Person:

def \_\_init\_\_(self, name):

self.name = name

def \_\_str\_\_(self):

return self.name

p1 = Person('amir')

print(p1)

خب حالا اگه کد بالا رو اجرا کنید نتیحه بهتری رو نمایش میده. در کد بالا به پایتون گفتیم زمانی که یک آبجکت رو چاپ کردیم مقدار name رو نشون بده:

amir

حالا متد repr چیه؟ متد repr هم دقیقا زمانی استفاده میشه که شما یک آبجکت رو صدا بزنید. اما تفاوتی که داره اینکه اگه کدتون رو از طریق ترمینال اجرا کنید متد repr فعال خواهد شد. درواقع متد repr برای زمانی هستش که شما قراره یک نتیجه رو به برنامه نویس دیگه نشون بدید اما متد str برای زمانی هستش که شما میخواید یک نتیجه رو به کاربر نشون بدین.

برای درک این موضوع داخل ترمنیال، با استفاده از دستور python وارد مفسر پایتون بشید.

حالا کد زیر رو وارد میکنیم:

>>> import datetime

>>> now = datetime.datetime.now()

>>> str(now)

'2019-8-15 10:29:34.786394'

>>> repr(now)

datetime.datetime(2019, 8, 15, 10, 29, 34, 786394)

در کد بالا ما زمان الآن رو ریختیم داخل متغیر now.

زمانی که با استفاده از متد str متغیر now رو صدا میزنیم میبینید که نتیجه به شکل یک زمان نشون داده میشه.

اما زمانی که متغیر now رو با متد repr صدا میزنیم نتیجه رو به شکل یک فانکشن میاره.

تفاوت این دو متد اینجا مشخص میشه. متد str برای نشون دادن یک نتیجه به کاربر عادی اما متد repr با هدف دیباگ کردن برنامه و نشون دادن یک پیغام به برنامه نویس دیگه استفاده میشه.

به صورت دیفالت داخل ترمینال متد repr صدا زده می‌شود

=================================================