class constructors

✓ fundamental part of object oriented programming

✓ they trigger python's instantiation process.

2 steps ↓
1- instance ==creation==
2 - instance ==initialization==

| reminder |

class
- attr : storing data
- method : providing behavior

once we have a class to work with, start creating new [instances] or [objects] of that class. * every instance is an object, not the way around.

the tool responsible for running this instantiation process is $\boxed{\text{class constructor}}$

calling a class : someclass( )

↓

when we call a class like this, we are calling the class constructor :

کارش اینه ⟵ which creates, initializes and returns a new object by triggering python's internal instantiation process.

مهم: calling a class isn't the same thing as calling an instance of a class.

↓

we need .__call__( ) method to make the class's instance callable

understanding python's instantiation process

when is it triggred? whenever we call a python class to create a new instance.

the process : 1- create a new instance

2. initialize the new instance with an appropriate initial state.

reminder : the __init__() method takes the $\boxed{\text{new object}}$ as its first argument, $\boxed{\text{self}}$.

instantiation Process

$$\begin{tikzcd}
\text{call the class constructor} \arrow[r] & \text{triggers instance creator .\_\_new\_\_()} \arrow[r] & \text{triggers instance initializer .\_\_init\_\_()}
\end{tikzcd}$$

object initializatio with .__init__()

✓ almost all classes need a custom init().

✓ the purpose of overriding init() is to give the objects a valid state so that we can use them in the code.

Learning the basics of writing our init methods.

✓ the most basic init: assigning input arguments to matching instance attributes.

Ex)
```
class name:
    def _init_(self, name):
        self.name = name
```

self: holds the new instance that results from calling .__new__().

نکته: the arguments to init() are the ones we passed in the call to the class constructor. (without considering self).

So, the init() signature defines the signature of the class constructor.

در init() نباید return تعریف کنیم درواقع return init باید none باشه نکته *

وگرنه type error برخورد میکنیم. زمانی هم return تعریف نکنیم اتوماتیک None برمیگردد در python.

برای init میتونیم condition تعریف کنیم و با استفاده از اون validate کنیم. نکته *

دیتای input را.

Building flexible object initializers

optional arguments: we can accept different sets of input arguments at instantiation time.

for example:
```
def __init__(self, name, formal=False).
```

Object Creation with .__new__()

✓ most of the time, we don't override .new().

Providing custom .__new__():

✓ only when you need to control the creation of a new instance at a low level.

follow these steps:

1) create a new instance by calling super().__new__().

2) customize the new instance

3) return the new instance.

نکته object.__new__( ) accepts only one argument

we shouldn't use *args and **kwargs when calling

$\text{super( ).\_\_new\_\_(cls, } \underbrace{\text{*args , **kwargs}}_{\text{delete this part}} \text{)}$

ادامه نکته → However, object.__new__( ) still accepts and passes over extra arguments to .__init__( ) if your class doesn't override .__new__( )

next step ⇒ some of the most common use cases of .new( ) in python programming.

*reminder: init کارهای set up را انجام میدهد.

Subclassing Immutable Built-in Types.

اینجا از __new__() استفاده می‌کنیم که بتونیم در زمان درست شدن (creation) چه تغییراتی روش اجرا بشه، چون ممکنه در زمان init دیر شده باشه و اون تغییرات را قبول نکنه. مثلاً: float در زمان init فقط value قبول می‌کنه ولی unit قبول نمی‌کنه، بنابراین ما __new__ را override می‌کنیم که unit هم قبول کنه.

→ it's too late to change it during initialization because the value is set during creation.

Returning Instances of another Class

وقتی می‌خوایم یه object از یه class دیگه صدا بزنیم از $\boxed{\text{new( )}}$ custom استفاده می‌کنیم.

نکته مهم: این جا دیگه هیچ وقت init، run نمی‌شه.

به خاطر این که این class هستیه یه object از classهای دیگه رو برمی‌گردونه و object کلاس خودشو برنمی‌گردونه که نیاز به initialization داشته باشه.

← مثال pet ما

Allowing only a Single Instance in Your Classes.

وقتی می‌خوایم singleton بسازیم یعنی یه کاری کنیم که کلاسی که تعریف می‌کنیم فقط یک instance داشته باشه، first instance, second instance یکی باشند و True برگردونه.

→ reminder: این ۳ تا کار موقع کار کردن با متد new( )

1) صدا زدن super.new( )

2) making change

3) returning

reminder → . __new__(cls) → به کلاس اشاره داره که یه instance از اون داره ساخته می‌شه.