The difference mainly arises with mutable vs immutable types.

\_\_new\_\_ accepts a *type* as the first argument, and (usually) returns a new instance of that type. Thus it is suitable for use with both mutable and immutable types.

\_\_init\_\_ accepts an *instance* as the first argument and modifies the attributes of that instance. This is inappropriate for an immutable type, as it would allow them to be modified after creation by calling obj.\_\_init\_\_(\*args).

Compare the behaviour of tuple and list:

>>> x = (1, 2)

>>> x

(1, 2)

>>> x.\_\_init\_\_([3, 4])

>>> x # tuple.\_\_init\_\_ does nothing

(1, 2)

>>> y = [1, 2]

>>> y

[1, 2]

>>> y.\_\_init\_\_([3, 4])

>>> y # list.\_\_init\_\_ reinitialises the object

[3, 4]

Whenever a class is instantiated \_\_new\_\_ and \_\_init\_\_ methods are called. \_\_new\_\_ method will be called when an object is created and \_\_init\_\_ method will be called to initialize the object. In the base class object, the \_\_new\_\_ method is defined as a static method which requires to pass a parameter cls. cls represents the class that is needed to be instantiated, and the compiler automatically provides this parameter at the time of instantiation.