**Interning** is an optimization technique by making global cache of particular objects in memory as they are instantiated. It is basically reusing objects on-demand.

In Python optimization, integer interning is the method of caching or pre-loading a list of integers at start up.

Standard implementation of Python (CPython) pre-loads (caches) a global list of **integers in the range from -5 to 256**. Any time an integer is referenced in this range Python does not create new one but uses the cached version. This is known as **integer interning**.

## **Why Does Python Intern?**

## This is basically an optimization technique in Python. Since small integers are relatively used more in our code than large integers. So, Python decides to pre-cache certain range (-5 to 256) of integers for performance reason.

### **Why not to intern everything?**

### **Answer** is performance & memory overhead. Interning everything requires a lot of memory & start up may take long since interning occurs at start up time.

So, question of what to intern & what not to intern is kind of trade-off. We choose that range of an integer in interning which has higher reusability.

Integers from -5 to 256 are known as **singleton objects** (classes that can be instantiated only once) in standard implementation of Python.

**String interning** is an optimization technique by making global cache of particular strings in memory as they are instantiated. It is basically reusing strings on-demand.

#### In Python optimization, string interning is the process of making cache of some strings to reduce memory use and to speed up processing.

In standard Python, following things are automatically interned:

1. As the Python code is compiled; identifiers — name given to variables, functions, classes, constants etc — are automatically interned. An identifier must start with \_ (underscore) or a letter and can only contain \_ , letters and numbers.
2. Some string literals that looks like an identifiers are also automatically interned.

## When To Use Manual String Interning

## In general, don't do manual string interning unless and until you have some valid reason to do it. It is becuase Python does this automatically in relevant cases. However, there are some cases where manual string interning is quite applicable:

1. Dealing with large number of strings where string has high frequency of repetition. This scenario generally occurs while tokenizing string in Natural Language Processing.
2. In those cases where strings need to be compared very often. Becuase string comparison using operator is (address comparison) is very faster than operator == (character wise comparison).

Ref: <https://www.codesansar.com/python-programming/string-interning.htm>