Type Casting:

Type casting means converting one data type into another.

Python supports TWO type of type casting--

1.Implicit Type Casting (Type Conversion) → Done automatically by Python.

2.Explicit Type Casting (Type Casting) → Done manually by the programmer.

1.Implicit Type Casting

Python automatically converts smaller data types into larger data types to avoid data loss.

In [1]:
```python
a = 5        # int
b = 2.5      # float

result = a + b   # int + float → float
print(result)    # 7.5
print(type(result))  # <class 'float'>
```

```
7.5
<class 'float'>
```

In [2]:
```python
x = 10
y = str(x)   # int → str
print(y, type(y))   # "10" <class 'str'>

z = int("20")   # str → int
print(z, type(z))   # 20 <class 'int'>
```

```
10 <class 'str'>
20 <class 'int'>
```

In [3]:
```python
a = 9.8
b = int(a)   # float → int (decimal part removed)
print(b)     # 9

c = float(7) # int → float
print(c)     # 7.0
```

```
9
7.0
```

In [4]:
```python
# List ↔ Tuple
nums = [1, 2, 3]
nums_tuple = tuple(nums)
print(nums_tuple)   # (1, 2, 3)

# String → List
s = "hello"
letters = list(s)
print(letters)  # ['h', 'e', 'l', 'l', 'o']

# List → Set
```

```
nums_set = set(nums)
print(nums_set)  # {1, 2, 3}
```

```
(1, 2, 3)
['h', 'e', 'l', 'l', 'o']
{1, 2, 3}
```

In [5]:

```
print(bool(0))      # False
print(bool(""))     # False
print(bool([]))     # False
print(bool(123))    # True
print(bool("Hi"))   # True
```

```
False
False
False
True
True
```

id() Function in Python:

- The id() function in Python returns the unique identity (memory address) of an object.
- Each object in Python has an id.
- This id is unique and constant for the object during its lifetime.

In [6]:

```
x = 10
y = 10
z = [10]

print(id(x))   # memory address of int 10
print(id(y))   # same as x, because Python caches small integers
print(id(z))   # different, since list is a new object
```

```
140713478669512
140713478669512
1693501029312
```

Tips:

1. Immutable objects (like int, str, tuple) → if two variables store the same value, they may share the same id (due to interning/caching).

In [7]:

```
a = "hello"
b = "hello"
print(id(a) == id(b))  # True
```

```
True
```

2. Mutable objects (like list, dict, set) → even if they look the same, different objects have different ids.

In [8]:

```
l1 = [1,2,3]
l2 = [1,2,3]
print(id(l1) == id(l2))  # False
```

```
False
```

3. is vs ==

- == checks value equality
- is checks object identity (id)