Type Casting techniques in python

The process of converting one possible type of value into another possible type of value is called "Type Casting".

In python programming, we have 5-fundamental type casting techniques.

int()

float()

bool()

complex()

str()

In [ ]:

int()

This function is used for converting one type of possible value into int type value.

syntax: varname: int(float / bool / comp / str)

In [2]:
```
a = 12.34
print(a,type(a))
print("-----------------")
b = int(a)
print(b,type(b))
```

```
12.34 <class 'float'>
-----------------
12 <class 'int'>
```

In [3]:
```
b = 24.56
print(b,type(b))
print("----------------------")
c = int(b)
print(c,type(c))
```

```
24.56 <class 'float'>
----------------------
24 <class 'int'>
```

In [4]:
```
a = True
print(a,type(a))
print("----------------------")
b = int(a)
print(b,type(b))
```

```
True <class 'bool'>
----------------------
1 <class 'int'>
```

In [5]:
```
a = False
print(a,type(a))
print("----------------------")
b = int(a)
print(b,type(b))
```

```
False <class 'bool'>
----------------------
0 <class 'int'>
```

In [6]:
```
a = 2 +3j # not possible
print(a,type(a))
print("--------------------")
b = int(a)
print(b,type(b))
```

```
(2+3j) <class 'complex'>
--------------------
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 4
      2 print(a,type(a))
      3 print("--------------------")
----> 4 b = int(a)
      5 print(b,type(b))

TypeError: int() argument must be a string, a bytes-like object or a real number,
not 'complex'
```

In [2]:
```
a = '12345'
print(a,type(a),id(a))
print("----------------------")
b = int(a)
print(b,type(b),id(b))
```

```
12345 <class 'str'> 1605914979776
----------------------
12345 <class 'int'> 1605914452400
```

In [ ]:
```
```

float

It is used for converting one type of possible value into float type value.

syntax: varname = float(int / bool/ complex / str)

In [3]:
```
a = 12
print(a,type(a),id(a))
print("---------------------")
b = float(a)
print(b,type(b),id(b))
```

```
12 <class 'int'> 140718899542808
---------------------
12.0 <class 'float'> 1605888662992
```

In [4]:
```
a = True
print(a,type(a),id(a))
print("---------------------")
b = float(a)
print(b,type(b),id(b))
```

```
True <class 'bool'> 140718898416512
---------------------
1.0 <class 'float'> 1605888666000
```

In [5]:
```
a = 2 + 3j # Value error
print(a,type(a),id(a))
print("---------------------")
b = float(a)
print(b,type(b),id(b))
```

```
(2+3j) <class 'complex'> 1605914452272
---------------------
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[5], line 4
      2 print(a,type(a),id(a))
      3 print("---------------------")
----> 4 b = float(a)
      5 print(b,type(b),id(b))

TypeError: float() argument must be a string or a real number, not 'complex'
```

In [6]:
```
a = "12"
print(a,type(a),id(a))
print("---------------------")
b = float(a)
print(b,type(b),id(b))
```

```
12 <class 'str'> 1605915164128
---------------------
12.0 <class 'float'> 1605888662992
```

In [7]:
```
a = "22.35"
print(a,type(a),id(a))
print("---------------------")
b = float(a)
print(b,type(b),id(b))
```

```
22.35 <class 'str'> 1605914745696
---------------------
22.35 <class 'float'> 1605914445360
```

In [ ]:
```

```

bool

It is used for converting one type of possible value into bool type of value

syntax: varname = bool(int\float\complex\str)

In [8]:
```
a = 100
print(a,type(a),id(a))
print("------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
100 <class 'int'> 140718899545624
------------------------
True <class 'bool'> 140718898416512
```

In [9]:
```
a = -120
print(a,type(a),id(a))
print("------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
-120 <class 'int'> 1605920330832
------------------------
True <class 'bool'> 140718898416512
```

In [10]:
```
a = 0000
print(a,type(a),id(a))
print("------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
0 <class 'int'> 140718899542424
------------------------
False <class 'bool'> 140718898416544
```

In [11]:
```
a = 1.4
print(a,type(a),id(a))
print("------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
1.4 <class 'float'> 1605914444976
------------------------
True <class 'bool'> 140718898416512
```

In [12]:
```
a = 0.0
print(a,type(a),id(a))
print("------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
0.0 <class 'float'> 1605890520560
------------------------
False <class 'bool'> 140718898416544
```

In [13]:
```
a = 2 + 3j
print(a,type(a),id(a))
print("------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
(2+3j) <class 'complex'> 1605920337680
------------------------
True <class 'bool'> 140718898416512
```

In [14]:
```
a = 0 + 0j
print(a,type(a),id(a))
print("------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
0j <class 'complex'> 1605920337136
------------------------
False <class 'bool'> 140718898416544
```

All Non-zero values are treated as True

Alll Zero values are treated as False

In [15]:
```
a = 0.0000000000001
print(a,type(a))
```

```
1e-13 <class 'float'>
```

In [16]:
```
a = 0.000000000000
print(a,type(a))
```

```
0.0 <class 'float'>
```

In [17]:
```
a = 0.0000000000001
print(a,type(a))
print("--------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
1e-13 <class 'float'>
--------------------------
True <class 'bool'> 140718898416512
```

In [18]:
```
a = 0.000000000000
print(a,type(a))
print("--------------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
0.0 <class 'float'>
--------------------------
False <class 'bool'> 140718898416544
```

In [19]:
```
a = "1234"
print(a,type(a))
print("--------------------")
```

```
b = bool(a)
print(b,type(b),id(b))
```

```
1234 <class 'str'>
--------------------
True <class 'bool'> 140718898416512
```

In [20]:
```
a = "4378"
print(a,type(a))
print("--------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
4378 <class 'str'>
--------------------
True <class 'bool'> 140718898416512
```

In [21]:
```
a = "0" # value is non-zero but len of str is 1
print(a,type(a))
print("--------------------")
b = bool(a)
print(b,type(b),id(b))
```

```
0 <class 'str'>
--------------------
True <class 'bool'> 140718898416512
```

In [ ]:
```

```

complex

It is used for converting one type of possible value into another complex type value.

Syntax: varname = complex(int\float\bool\str)

In [23]:
```
a = 10
print(a,type(a),id(a))
print("-----------------------")
b = complex(a)
print(b,type(b),id(b))
```

```
10 <class 'int'> 140718899542744
-----------------------
(10+0j) <class 'complex'> 1605920325840
```

In [24]:
```
a = 1.7
print(a,type(a),id(a))
print("-----------------------")
b = complex(a)
print(b,type(b),id(b))
```

```
1.7 <class 'float'> 1605888666288
-----------------------
(1.7+0j) <class 'complex'> 1605920291152
```

In [25]:
```
a = True
print(a,type(a),id(a))
print("-----------------------")
b = complex(a)
print(b,type(b),id(b))
```

```
True <class 'bool'> 140718898416512
-----------------------
(1+0j) <class 'complex'> 1605920337584
```

In [26]:
```
a = False
print(a,type(a),id(a))
print("-----------------------")
b = complex(a)
print(b,type(b),id(b))
```

```
False <class 'bool'> 140718898416544
-----------------------
0j <class 'complex'> 1605914452560
```

In [27]:
```
a = "18"
print(a,type(a),id(a))
print("-----------------------")
b = complex(a)
print(b,type(b),id(b))
```

```
18 <class 'str'> 1605916329424
-----------------------
(18+0j) <class 'complex'> 1605920282704
```

In [ ]:
```

```