Type Conversion in Python

Python defines type conversion functions to directly convert one data type to another which is useful in day-to-day and competitive programming. This article is aimed at providing information about certain conversion functions.

There are two types of Type Conversion in Python:

1. Implicit Type Conversion
2. Explicit Type Conversion

Implicit Type Conversion

In Implicit type conversion of data types in Python, the Python interpreter automatically converts one data type to another without any user involvement.

EXAMPLE:

```
x = 10
print("x is of type:",type(x))

y = 10.6
print("y is of type:",type(y))

z = x + y

print(z)
print("z is of type:",type(z))
```

OUTPUT:

```
x is of type: <class 'int'>
y is of type: <class 'float'>
20.6
z is of type: <class 'float'>
```

As we can see the data type of ‘z’ got automatically changed to the “float” type while one variable x is of integer type while the other variable y is of float type. The reason for the float value not being converted into an integer instead is due to type promotion that allows performing operations by converting data into a wider-sized data type without any loss of information. This is a simple case of Implicit type conversion in python.

Explicit Type Conversion

In Explicit Type Conversion in Python, the data type is manually changed by the user as per their requirement. With explicit type conversion, there is a risk of data loss since we are forcing an expression to be changed in some specific data type. Various forms of explicit type conversion are explained below:

1. int(a, base): This function converts any data type to integer. ‘Base’ specifies the base in which string is if the data type is a string.

2. float(): This function is used to convert any data type to a floating-point number.

```python
# Python code to demonstrate Type conversion using int(), float()

# initializing string
s = "10010"

# printing string converting to int base 2
c = int(s,2)
print ("After converting to integer base 2: ", end="")
print (c)

# printing string converting to float
e = float(s)
print ("After converting to float: ", end="")
print (e)
```

```
After converting to integer base 2: 18
After converting to float: 10010.0
```

3. tuple() : This function is used to convert to a tuple.

4. set() : This function returns the type after converting to set.

5. list() : This function is used to convert any data type to a list type.

```
# Python code to demonstrate Type conversion
# using  tuple(), set(), list()

# initializing string
s = 'geeks'

# printing string converting to tuple
c = tuple(s)
print ("After converting string to a tuple: ",end="")
print (c)

# printing string converting to set
c = set(s)
print ("After converting string to set: ",end="")
print (c)

# printing string converting to a list
c = list(s)
print ("After converting string to the list: ",end="")
print (c)
```

```
After converting string to tuple: ('g', 'e', 'e', 'k', 's')
After converting string to set: {'s', 'e', 'g', 'k'}
After converting string to list: ['g', 'e', 'e', 'k', 's']
```

9. dict() : This function is used to convert a tuple of order (key,value) into a dictionary.

10. str() : Used to convert integer into a string.

11. complex(real,imag) : This function converts real numbers to complex(real,imag) number.

```
# Python code to demonstrate Type conversion
# using  dict(), complex(), str()

# initializing integers
a = 1
b = 2

# initializing tuple
tup = (('a', 1) ,('f', 2), ('g', 3))

# printing integer converting to complex number
c = complex(1,2)
print ("After converting an integer to complex number : ",end="")
print (c)

# printing integer converting to string
c = str(a)
print ("After converting an integer to string : ",end="")
print (c)

# printing tuple converting to expression dictionary
c = dict(tup)
print ("After converting tuple to the dictionary: ",end="")
print (c)
```

```
After converting integer to complex number : (1+2j)
After converting integer to string : 1
After converting tuple to dictionary : {'a': 1, 'f': 2, 'g': 3}
```

Note that type conversion might lead to errors if the input value cannot be converted to the desired data type. For example, trying to convert a non-numeric string to an integer or float will raise a ValueError. Similarly, converting a non-boolean string to a boolean might not give the expected result. Always make sure that the conversion is valid for your specific use case.