1 Introduction

In computer programming, data types specify the type of data that can be stored inside a variable. For example,

```
num = 24

```

Here, 24 (an integer) is assigned to the num variable. So the data type of num is of the int class.

2 Python Data Types

Data Types	Classes	Description
Numeric	int, float, complex	Holds numeric values
String	str	Holds a sequence of characters
Sequence	list, tuple, range	Holds a collection of items
Mapping	dict	Holds data in key-value pair form
Boolean	bool	Holds either True or False
Set	set, frozenset	Holds a collection of unique items

Table 1: Python Data Types

Since everything is an object in Python programming, data types are actually classes and variables are instances (objects) of these classes.

3 Python Numeric Data Type

In Python, the numeric data type is used to hold numeric values.

```
num1 = 5
print(num1, 'is of type', type(num1))

num2 = 2.0
print(num2, 'is of type', type(num2))

num3 = 1+2j
print(num3, 'is of type', type(num3))

```

Output:

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is of type <class 'complex'>

```

In the above example, we have created three variables named num1, num2, and num3 with values 5, 5.0, and 1+2j respectively.

4 Python List Data Type

List is an ordered collection of similar or different types of items separated by commas and enclosed within brackets `[]`. For example,

```
  languages = ["Swift", "Java", "Python"]

```

...

5 Python List Data Type (Continued)

List is an ordered collection of similar or different types of items separated by commas and enclosed within brackets `[]`. For example,

```
languages = ["Swift", "Java", "Python"]
```

5.1 Access List Items

To access items from a list, we use the index number (0, 1, 2, ...). For example,

```
languages = ["Swift", "Java", "Python"]

% access element at index 0
print(languages[0])   % Swift

% access element at index 2
print(languages[2])   % Python
```

In the above example, we have used the index values to access items from the `languages` list.

6 Python Tuple Data Type

Tuple is an ordered sequence of items same as a list. The only difference is that tuples are immutable. Tuples once created cannot be modified.

In Python, we use the parentheses `()` to store items of a tuple. For example,

```
product = ('Xbox', 499.99)
```

6.1 Access Tuple Items

Similar to lists, we use the index number to access tuple items in Python. For example,

```
% create a tuple
product = ('Microsoft', 'Xbox', 499.99)

% access element at index 0
print(product[0])   % Microsoft

% access element at index 1
print(product[1])   % Xbox
```

7 Python String Data Type

String is a sequence of characters represented by either single or double quotes. For example,

```
name = 'Python'
print(name)

message = 'Python for beginners'
print(message)
```

8 Python Set Data Type

Set is an unordered collection of unique items. Set is defined by values separated by commas inside braces {}. For example,

```
% create a set named student_id
student_id = {112, 114, 116, 118, 115}

% display student_id elements
print(student_id)

% display type of student_id
print(type(student_id))
```

9 Python Dictionary Data Type

Python dictionary is an ordered collection of items. It stores elements in key/-value pairs.

9.1 Access Dictionary Values Using Keys

We use keys to retrieve the respective value. But not the other way around. For example,

```
% create a dictionary named capital_city
capital_city = {'Nepal': 'Kathmandu', 'Italy': 'Rome', 'England':
   'London'}

print(capital_city['Nepal'])  % prints Kathmandu

% Throws error message
print(capital_city['Kathmandu'])
```

10 Python Type Conversion

In programming, type conversion is the process of converting data of one type to another. There are two types of type conversion in Python.

10.1 Implicit Conversion

In certain situations, Python automatically converts one data type to another. This is known as implicit type conversion.

10.1.1 Example 1: Converting integer to float

Let's see an example where Python promotes the conversion of the lower data type (integer) to the higher data type (float) to avoid data loss.

```
integer_number = 123
float_number = 1.23

new_number = integer_number + float_number

% display new value and resulting data type
print("Value:",new_number)
print("Data Type:",type(new_number))
```

Output:

```
Value: 124.23
Data Type: <class 'float'>
```

In the above example, we have created two variables: `integer_number` and `float_number` of `int` and `float` type respectively.

Then we added these two variables and stored the result in `new_number`.

As we can see, `new_number` has the value 124.23 and is of the `float` data type.

10.1.2 Note

We get `TypeError`, if we try to add `str` and `int`. For example, '12' + 23. Python is not able to use Implicit Conversion in such conditions.

10.2 Explicit Conversion

In Explicit Type Conversion, users convert the data type of an object to the required data type.

We use the built-in functions like `int()`, `float()`, `str()`, etc to perform explicit type conversion.

This type of conversion is also called typecasting because the user casts (changes) the data type of the objects.

10.2.1 Example 2: Addition of string and integer Using Explicit Conversion

```
num_string = '12'
num_integer = 23

print("Data type of num_string before Type Casting:",type(
  num_string))
```

```

% explicit type conversion
num_string = int(num_string)

print("Data type of num_string after Type Casting:",type(
  num_string))

num_sum = num_integer + num_string

print("Sum:",num_sum)
print("Data type of num_sum:",type(num_sum))
```

Output:

```
Data type of num_string before Type Casting: <class 'str'>
Data type of num_string after Type Casting: <class 'int'>
Sum: 35
Data type of num_sum: <class 'int'>
```

In the above example, we have created two variables: `num_string` and `num_integer` with `str` and `int` type values respectively. Notice the code,

`num_string = int(num_string)`

Here, we have used `int()` to perform explicit type conversion of `num_string` to integer type.

After converting `num_string` to an integer value, Python is able to add these two variables.

Finally, we got the `num_sum` value i.e 35 and data type to be `int`.

10.3 Key Points to Remember

- Type Conversion is the conversion of an object from one data type to another.
- Implicit Type Conversion is automatically performed by the Python interpreter.
- Python avoids the loss of data in Implicit Type Conversion.
- Explicit Type Conversion is also called Type Casting, the data types of objects are converted using predefined functions by the user.
- In Type Casting, loss of data may occur as we enforce the object to a specific data type.